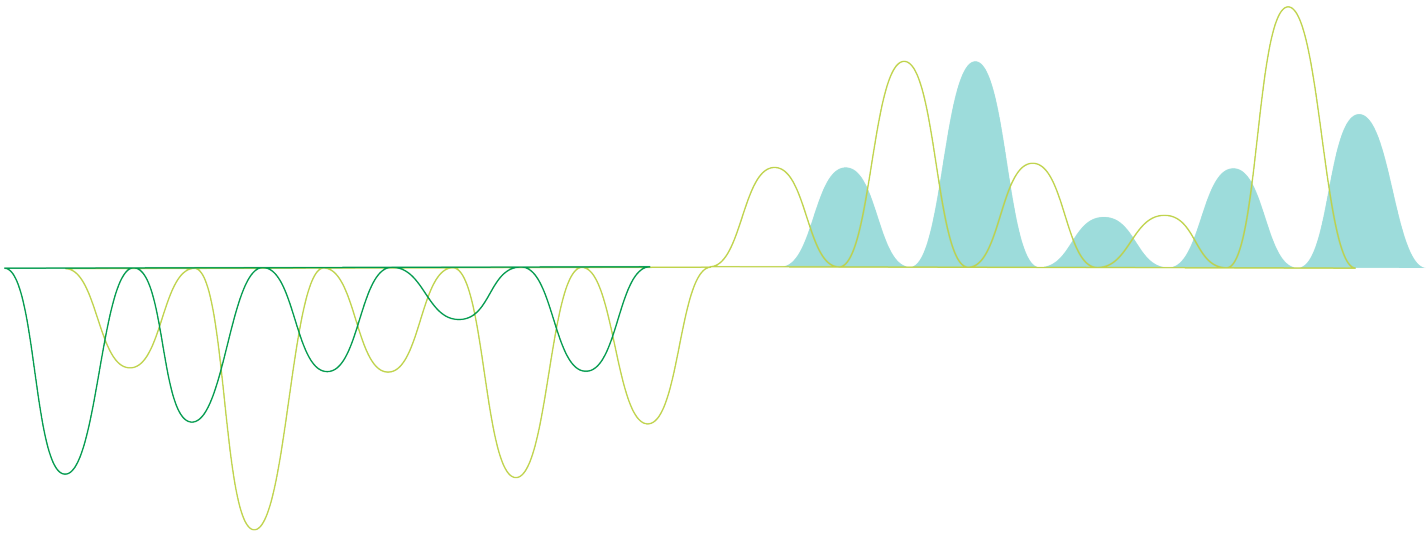
Skriptsyntax und Diagrammfunktionen

Qlik Sense®
February 2024
Copyright © 1993-jjjj QlikTech International AB. Alle Rechte vorbehalten.



© jjjj QlikTech International AB. All rights reserved. All company and/or product names may be trade names, trademarks and/or registered trademarks of the respective owners with which they are associated.

Contents

1 Was ist Qlik Sense? 16
1.1 Was ist in Qlik Sense möglich? 16
1.2 Wie funktioniert Qlik Sense? 16
Das App-Modell 16
Die assoziativen Möglichkeiten 16
Zusammenarbeit und Mobilität 16
1.3 Bereitstellungsmöglichkeiten für Qlik Sense 17
Qlik Sense Desktop 17
Qlik Sense Enterprise 17
1.4 Administration und Verwaltung einer Qlik Sense-Site 17
1.5 Erweiterung von Qlik Sense und individuelle Anpassung 17
Erstellung von Erweiterungen und Mashups 17
Erstellung von Clients 17
Erstellung von Servertools 17
Herstellung von Verbindungen zu anderen Datenquellen 17
2 Skript-Syntax – Überblick 18
2.1 Einführung in Skriptsyntax 18
2.2 Was ist der Backus-Naur-Formalismus? 18
3 Skriptbefehle 20
3.1 Steuerungsbefehle im Skript 20
Steuerungsbefehle im Skript – Übersicht 20
Call 22
Do..loop 23
End 24
Exit 24
Exit script 25
For..next 25
For each..next 27
If..then..elseif..else..end if 30
Next 31
Sub..end sub 31
Switch..case..default..end switch 33
To 34
3.2 Skript-Zusätze 34
Skript-Zusätze – Übersicht 34
Add 38
Buffer 40
Concatenate 42
Crosstable 47
First 57
Generic 59
Hierarchy 66
HierarchyBelongsTo 68
Inner 70
IntervalMatch 71
Join 75
Keep 84

Left ... 85
Mapping ... 86
Merge ... 88
NoConcatenate ... 93
Only ... 102
Outer ... 102
Partielles Laden ... 103
Replace ... 107
Right ... 108
Sample ... 109
Semantic ... 113
Unless ... 117
When ... 122
3.3 Reguläre Skriptbefehle ... 129
Reguläre Skriptbefehle – Übersicht ... 129
Alias ... 135
AutoNumber ... 136
Binary ... 139
Comment field ... 141
Comment table ... 142
Connect ... 142
Declare ... 144
Derive ... 147
Direct Query ... 148
Directory ... 153
Disconnect ... 154
Drop ... 154
Drop table ... 155
Execute ... 156
Field/Fields ... 157
FlushLog ... 157
Force ... 158
From ... 160
Load ... 160
Let ... 179
Loosen Table ... 180
Map ... 180
NullAsNull ... 181
NullAsValue ... 182
Qualify ... 183
Rem ... 184
Rename ... 185
Search ... 186
Section ... 187
Select ... 188
Set ... 190
Sleep ... 191
SQL ... 191

SQLColumns ... 192
SQLTables ... 193
SQLTypes ... 193
Star ... 194
Store ... 196
Table/Tables ... 201
Tag ... 201
Trace ... 202
Unmap ... 202
Unqualify ... 203
Untag ... 204
3.4 Arbeitsverzeichnis ... 204
Qlik Sense Desktop-Arbeitsverzeichnis ... 205
Qlik Sense-Arbeitsverzeichnis ... 205
4 Arbeiten mit Variablen im Dateneditor ... 206
4.1 Überblick ... 206
4.2 Festlegen einer Variable ... 206
4.3 Löschen einer Variablen ... 207
4.4 Laden eines Variablenwerts als Feldwert ... 207
4.5 Variable Berechnung ... 207
4.6 Systemvariablen ... 208
Systemvariablen – Übersicht ... 208
CreateSearchIndexOnReload ... 211
HidePrefix ... 211
HideSuffix ... 212
Include ... 212
OpenUrlTimeout ... 213
StripComments ... 214
Verbatim ... 214
4.7 Variablen zur Verarbeitung der Werte ... 214
Variablen zur Verarbeitung der Werte – Übersicht ... 214
NullDisplay ... 215
NullInterpret ... 215
NullValue ... 216
OtherSymbol ... 216
4.8 Variablen zur Interpretation von Zahlen ... 216
Währungsformat ... 217
Zahlenformat ... 217
Zeitformat ... 217
BrokenWeeks ... 219
DateFormat ... 220
DayNames ... 226
DecimalSep ... 231
FirstWeekDay ... 233
LongDayNames ... 238
LongMonthNames ... 241
MoneyDecimalSep ... 245

MoneyFormat 249
MoneyThousandSep 253
MonthNames 258
NumericalAbbreviation 263
ReferenceDay 264
ThousandSep 269
TimeFormat 275
TimestampFormat 275
4.9 Direct Discovery-Variablen 278
Direct Discovery-Systemvariablen 278
Teradata Query-Banding-Variablen 280
Direct Discovery-Zeichenvariablen 280
Variable zur Interpretation von Direct Discovery-Zahlen 281
4.10 Fehlervariablen 282
Fehlervariablen – Übersicht 282
ErrorMode 283
ScriptError 284
ScriptErrorCount 285
ScriptErrorList 285
5 Formeln im Skript 286
6 Diagrammformeln 288
6.1 Definieren des Aggregierungsbereichs 288
6.2 Aggregierung mit Auswahlformeln 291
Auswahlformeln 291
Beispiele 292
Natürliche Sätze 292
Identifikatoren für Auswahlformeln 295
Operatoren für Auswahlformeln 296
Modifikatoren für Auswahlformeln 297
Innere und äußere Auswahlformeln 320
Tutorial – Erstellen einer Auswahlformel 322
Syntax für Auswahlformeln 332
6.3 Allgemeine Syntax für Diagrammformeln 332
6.4 Allgemeine Syntax für Aggregierungsformeln 333
7 Operatoren 334
7.1 Bit-Operatoren 334
7.2 Logische Operatoren 335
7.3 Numerische Operatoren 335
7.4 Relationale Operatoren 336
7.5 String-Operatoren 337
& 338
like 338
8 Skript- und Diagrammfunktionen 339
8.1 Analyseverbindungen für serverseitige Erweiterungen (SSE) 339
8.2 Aggregierungsfunktionen 339
Aggregierungsfunktionen im Datenladeskript verwenden 340

Aggregierungsfunktionen in Diagrammformeln verwenden ... 340
Berechnung von Aggregierungen ... 340
Aggregierung von Schlüsselfeldern ... 340
Einfache Aggregierungsfunktionen ... 341
Aggregierung von Häufigkeiten ... 365
Finanz-Aggregierung ... 383
Statistische Aggregierungsfunktionen ... 411
Funktionen für statistische Tests ... 482
String-Aggregierungsfunktionen ... 549
Funktionen für synthetische Dimensionen ... 562
Verschachtelte Aggregierung ... 565
8.3 Aggr - Diagrammfunktion ... 566
Beispiele: Diagrammformeln mit Aggr ... 569
8.4 Farbfunktionen ... 572
Vordefinierte Farbfunktionen ... 574
ARGB ... 575
RGB ... 576
HSL ... 578
8.5 Konditionalfunktionen ... 579
Konditionalfunktionen – Übersicht ... 579
alt ... 580
class ... 581
coalesce ... 582
if ... 583
match ... 587
mixmatch ... 590
pick ... 593
wildmatch ... 593
8.6 Counter-Funktionen ... 596
Counter-Funktionen – Übersicht ... 596
autonumber ... 597
autonumberhash128 ... 600
autonumberhash256 ... 602
IterNo ... 604
RecNo ... 605
RowNo ... 606
RowNo - Diagrammfunktion ... 607
8.7 Funktionen für Datum und Uhrzeit ... 610
Funktionen für Datum und Uhrzeit – Übersicht ... 610
addmonths ... 619
addyears ... 629
age ... 636
converttolocaltime ... 638
day ... 642
dayend ... 648
daylightsaving ... 656
dayname ... 657
daynumberofquarter ... 659

daynumberofyear 666
daystart 672
firstworkdate 679
GMT 681
hour 685
inday 689
indaytotime 697
inlunarweek 707
inlunarweektodate 720
inmonth 731
inmonths 739
inmonthstodate 753
inmonthtodate 766
inquarter 776
inquartertodate 790
inweek 803
inweektodate 819
inyear 833
inyeartodate 846
lastworkdate 859
localtime 868
lunarweekend 872
lunarweekname 884
lunarweekstart 896
makedate 908
maketime 915
makeweekdate 922
minute 931
month 936
monthend 943
monthname 952
monthsend 960
monthsname 973
monthsstart 987
monthstart 1000
networkdays 1010
now 1020
quarterend 1027
quartername 1041
quarterstart 1053
second 1065
setdateyear 1070
setdateyearmonth 1072
timezone 1074
today 1074
UTC 1080
week 1081
weekday 1097

weekend 1106
weekname 1118
weekstart 1133
weekyear 1146
year 1155
yearend 1162
yearname 1174
yearstart 1187
yeartodate 1199
8.8 Exponential- und Logarithmusfunktionen 1215
8.9 Feldfunktionen 1216
Counter-Funktionen 1216
Feld- und Auswahlfunktionen 1217
GetAlternativeCount - Diagrammfunktion 1217
GetCurrentSelections - Diagrammfunktion 1218
GetExcludedCount - Diagrammfunktion 1220
GetFieldSelections - Diagrammfunktion 1221
GetNotSelectedCount - Diagrammfunktion 1224
GetObjectDimension - Diagrammfunktion 1224
GetObjectField - Diagrammfunktion 1225
GetObjectMeasure - Diagrammfunktion 1226
GetPossibleCount - Diagrammfunktion 1226
GetSelectedCount - Diagrammfunktion 1228
8.10 Dateifunktionen 1229
Dateifunktionen – Übersicht 1229
Attribute 1231
ConnectString 1239
FileBaseName 1240
FileDir 1240
FileExtension 1241
FileName 1241
FilePath 1241
FileSize 1242
FileTime 1243
GetFolderPath 1244
QvdCreateTime 1245
QvdFieldName 1246
QvdNoOfFields 1246
QvdNoOfRecords 1247
QvdTableName 1248
8.11 Finanzfunktionen 1249
Finanzfunktionen – Übersicht 1250
BlackAndSchole 1250
FV 1251
nPer 1252
Pmt 1253
PV 1254
Rate 1255

8.12 Formatfunktionen ... 1256
Formatfunktionen – Übersicht ... 1256
ApplyCodepage ... 1257
Date ... 1258
Dual ... 1260
Interval ... 1262
Money ... 1263
Num ... 1264
Time ... 1267
Timestamp ... 1268
8.13 Numerische Funktionen ... 1269
Numerische Funktionen – Übersicht ... 1270
Kombinations- und Variationsfunktionen ... 1270
Modulo-Funktionen ... 1271
Paritätsfunktionen ... 1271
Rundungsfunktionen ... 1271
BitCount ... 1272
Ceil ... 1272
Combin ... 1273
Div ... 1274
Even ... 1274
Fabs ... 1275
Fact ... 1275
Floor ... 1276
Fmod ... 1277
Frac ... 1278
Mod ... 1278
Odd ... 1279
Permut ... 1279
Round ... 1280
Sign ... 1282
8.14 Räumliche Funktionen ... 1282
Räumliche Funktionen – Übersicht ... 1282
GeoAggrGeometry ... 1284
GeoBoundingBox ... 1285
GeoCountVertex ... 1286
GeoGetBoundingBox ... 1286
GeoGetPolygonCenter ... 1287
GeoInvProjectGeometry ... 1287
GeoMakePoint ... 1288
GeoProject ... 1288
GeoProjectGeometry ... 1289
GeoReduceGeometry ... 1290
8.15 Interpretationsfunktionen ... 1291
Interpretationsfunktionen – Übersicht ... 1291
Date# ... 1293
Interval# ... 1294
Money# ... 1294

Num# 1296
Text 1296
Time# 1297
Timestamp# 1298
8.16 Inter-Record-Funktionen 1299
Zeilenfunktionen 1300
Spaltenfunktionen 1300
Feldfunktionen 1301
Pivottabellenfunktionen 1301
Inter-Record-Funktionen im Datenladeskript 1302
Above - Diagrammfunktion 1303
Below - Diagrammfunktion 1308
Bottom - Diagrammfunktion 1311
Column - Diagrammfunktion 1316
Dimensionality - Diagrammfunktion 1318
Exists 1319
FieldIndex 1323
FieldValue 1325
FieldValueCount 1326
LookUp 1328
NoOfRows - Diagrammfunktion 1330
Peek 1332
Previous 1340
Top - Diagrammfunktion 1341
SecondaryDimensionality - Diagrammfunktion 1346
After - Diagrammfunktion 1346
Before - Diagrammfunktion 1347
First - Diagrammfunktion 1348
Last - Diagrammfunktion 1350
ColumnNo - Diagrammfunktion 1351
NoOfColumns - Diagrammfunktion 1351
8.17 Logische Funktionen 1352
8.18 Mapping-Funktionen 1353
Mapping-Funktionen – Übersicht 1353
ApplyMap 1353
MapSubstring 1355
8.19 Mathematische Funktionen 1357
8.20 NULL-Funktionen 1358
NULL-Funktionen – Übersicht 1358
EmptyIsNull 1358
IsNull 1359
NULL 1360
8.21 Bereichsfunktionen 1361
Einfache Abschnittsfunktionen 1361
Abschnittsfunktionen für Counter 1362
Statistische Abschnittsfunktionen 1362
Finanz-Abschnittsfunktionen 1363
RangeAvg 1364

RangeCorrel ... 1366
RangeCount ... 1368
RangeFractile ... 1371
RangeIRR ... 1373
RangeKurtosis ... 1374
RangeMax ... 1375
RangeMaxString ... 1377
RangeMin ... 1379
RangeMinString ... 1381
RangeMissingCount ... 1382
RangeMode ... 1384
RangeNPV ... 1386
RangeNullCount ... 1387
RangeNumericCount ... 1389
RangeOnly ... 1390
RangeSkew ... 1391
RangeStdev ... 1392
RangeSum ... 1394
RangeTextCount ... 1396
RangeXIRR ... 1398
RangeXNPV ... 1399
8.22 Relationale Funktionen ... 1401
Rangfolgefunktionen ... 1402
Clustering-Funktionen ... 1402
Zeitreihenzerlegungs-Funktionen ... 1403
Rank - Diagrammfunktion ... 1404
HRank - Diagrammfunktion ... 1408
Optimieren mit k-means: Ein reales Beispiel ... 1410
KMeans2D - Diagrammfunktion ... 1419
KMeansND - Diagrammfunktion ... 1434
KMeansCentroid2D - Diagrammfunktion ... 1449
KMeansCentroidND - Diagrammfunktion ... 1450
STL_Trend - Diagrammfunktion ... 1451
STL_Seasonal - Diagrammfunktion ... 1453
STL_Residual - Diagrammfunktion ... 1455
Tutorial – Zeitreihenzerlegung in Qlik Sense ... 1457
8.23 Verteilungsfunktionen ... 1462
Verteilungsfunktionen für statistische Tests – Übersicht ... 1462
BetaDensity ... 1465
BetaDist ... 1465
BetaInv ... 1465
BinomDist ... 1466
BinomFrequency ... 1466
BinomInv ... 1467
ChiDensity ... 1467
ChiDist ... 1468
ChiInv ... 1468
FDensity ... 1469

FDist ... 1469
FInv ... 1470
GammaDensity ... 1471
GammaDist ... 1471
GammaInv ... 1472
NormDist ... 1472
NormInv ... 1473
PoissonDist ... 1474
PoissonFrequency ... 1474
PoissonInv ... 1474
TDensity ... 1475
TDist ... 1475
TInv ... 1476
8.24 Stringfunktionen ... 1477
Stringfunktionen – Übersicht ... 1477
Capitalize ... 1480
Chr ... 1481
Evaluate ... 1481
FindOneOf ... 1482
Hash128 ... 1483
Hash160 ... 1484
Hash256 ... 1485
Index ... 1486
IsJson ... 1487
JsonGet ... 1488
JsonSet ... 1489
KeepChar ... 1489
Left ... 1491
Len ... 1491
LevenshteinDist ... 1492
Lower ... 1493
LTrim ... 1494
Mid ... 1495
Ord ... 1496
PurgeChar ... 1496
Repeat ... 1497
Replace ... 1498
Right ... 1499
RTrim ... 1500
SubField ... 1501
SubStringCount ... 1504
TextBetween ... 1505
Trim ... 1506
Upper ... 1507
8.25 Systemfunktionen ... 1507
Systemfunktionen – Übersicht ... 1507
EngineVersion ... 1510
GetSysAttr ... 1510

InObject - Diagrammfunktion 1511
IsPartialReload 1515
ObjectId - Diagrammfunktion 1516
ProductVersion 1518
StateName - Diagrammfunktion 1519
8.26 Tabellenfunktionen 1519
Tabellenfunktionen – Übersicht 1519
FieldName 1521
FieldNumber 1522
NoOfFields 1522
NoOfRows 1523
8.27 Trigonometrische und hyperbolische Funktionen 1523
8.28 Window-Funktionen 1525
Window – Skriptfunktion 1526
WRank – Skriptfunktion 1534
9 Zugriffsbeschränkung für Dateisystem 1540
9.1 Sicherheitsaspekte beim Verbinden mit dateibasierten ODBC- und OLE DB-Datenverbindungen 1540
9.2 Einschränkungen im Standardmodus 1540
Systemvariablen 1541
Reguläre Skriptbefehle 1542
Steuerungsbefehle im Skript 1544
Dateifunktionen 1544
Systemfunktionen 1547
9.3 Deaktivieren des Standardmodus 1547
Qlik Sense 1547
Qlik Sense Desktop 1547
10 Skripterstellung auf Diagrammebene 1549
10.1 Steuerungsbefehle 1549
Steuerungsbefehle zur Diagrammänderung – Übersicht 1549
Call 1551
Do..loop 1552
End 1553
Exit 1553
Exit script 1553
For..next 1554
For each..next 1555
If..then..elseif..else..end if 1558
Next 1559
Sub..end sub 1559
Switch..case..default..end switch 1561
To 1562
10.2 Zusätze 1562
Zusätze zur Diagrammänderung – Übersicht 1562
Add 1562
Replace 1563
10.3 Reguläre Anweisungen 1563

Reguläre Anweisungen zur Diagrammänderung – Übersicht 1563
Load 1564
Let 1570
Set 1571
Put 1571
HCValue 1572
11 QlikView-Funktionen und -Befehle, die in Qlik Sense nicht unterstützt werden 1574
11.1 Skriptbefehle, die in Qlik Sense nicht unterstützt werden 1574
11.2 Funktionen, die in Qlik Sense nicht unterstützt werden 1574
11.3 Zusätze, die in Qlik Sense nicht unterstützt werden 1574
12 Funktionen und Befehle, die in Qlik Sense nicht empfohlen werden 1575
12.1 Skriptbefehle, die in Qlik Sense nicht empfohlen werden 1575
12.2 Parameter für Skriptbefehle, die in Qlik Sense nicht empfohlen werden 1575
12.3 Funktionen, die in Qlik Sense nicht empfohlen werden 1577
Qualifizierer ALL 1577

1 Was ist Qlik Sense?

Qlik Sense ist eine Plattform zur Datenanalyse. Mit Qlik Sense können Sie Daten analysieren und Erkenntnisse daraus gewinnen. Sie können Ihre Erkenntnisse auch mit anderen teilen und Daten in Gruppen und Unternehmen analysieren. Mit Qlik Sense können Sie Ihre eigenen Fragen stellen und beantworten und Ihre eigenen Rückschlüsse ziehen. Mit Qlik Sense können Sie und Ihre Kollegen gemeinsam Entscheidungen treffen.

1.1 Was ist in Qlik Sense möglich?

Mit den meisten Business Intelligence-Produkten (BI) können Sie Fragen beantworten, die von vornherein klar sind. Was ist jedoch mit den Fragen, die sich daraus ergeben? Die Fragen, die gestellt werden, nachdem jemand Ihren Bericht gelesen oder sich Ihre Visualisierungen angesehen hat? Mit den assoziativen Möglichkeiten in Qlik Sense können Sie Fragen der Reihe nach beantworten und Ihren eigenen Weg zum Ergebnis wählen. Mit Qlik Sense können Sie Ihre Daten mit nur einem Klick untersuchen, bei jedem Schritt Erkenntnisse gewinnen und den nächsten Schritt basierend auf den gewonnen Erkenntnissen bestimmen.

1.2 Wie funktioniert Qlik Sense?

Qlik Sense bietet verschiedene Anzeigemöglichkeiten für Ihre Daten. Qlik Sense erfordert keine vordefinierten statischen Berichte und Sie sind nicht von anderen Benutzern abhängig. Sie müssen einfach nur klicken und Ihre Schlüsse ziehen. Bei jedem Klick reagiert Qlik Sense sofort und aktualisiert jede Qlik Sense-Visualisierung und Ansicht in der App anhand der neu berechneten Daten und Visualisierungen, die von Ihren Auswahlen abhängen.

Das App-Modell

Sie müssen keine umfangreichen Geschäftsanwendungen bereitstellen und verwalten. Erstellen Sie einfach Ihre eigenen Qlik Sense-Apps, die Sie wiederverwenden, bearbeiten und für andere bereitstellen können. Mit dem App-Modell können Sie selbst die nächste Frage stellen und die Antwort finden, ohne einen Fachmann um einen neue Bericht oder eine Visualisierung bitten zu müssen.

Die assoziativen Möglichkeiten

Qlik Sense verwaltet automatisch alle Beziehungen zwischen den Daten und präsentiert die Informationen mithilfe eines Farbschemas in **green/white/gray**. Auswahlen werden in Grün, verknüpfte Daten in Weiß und ausgeschlossene (nicht verknüpfte) Daten in Grau dargestellt. Durch diese sofortige Rückmeldung ergeben sich neue Fragen und Sie können die Daten weiter untersuchen und Erkenntnisse daraus ziehen.

Zusammenarbeit und Mobilität

Über Qlik Sense können Sie außerdem mit Kollegen zusammenarbeiten, unabhängig davon, wo und in welcher Zeitzone sie sich befinden. Alle Qlik Sense-Funktionen einschließlich der Möglichkeiten zur Assoziation und Zusammenarbeit sind auch auf Mobilgeräten verfügbar. Mit Qlik Sense können Sie Ihre eigenen Fragen stellen und beantworten – und zwar unabhängig von Ihrem Standort auch mit Kollegen.

1.3 Bereitstellungsmöglichkeiten für Qlik Sense

Zwei Versionen von Qlik Sense können bereitgestellt werden: Qlik Sense Desktop und Qlik Sense Enterprise.

Qlik Sense Desktop

Hierbei handelt es sich um eine Version für Einzelbenutzer, die sich ganz einfach – normalerweise auf einem lokalen Computer – installieren lässt.

Qlik Sense Enterprise

Diese Version wird zur Bereitstellung von Qlik Sense-Sites genutzt. Eine Site besteht aus einer oder mehreren Servermaschinen, die mit einem gemeinsamen logischen Repository oder zentralen Knoten verbunden sind.

1.4 Administration und Verwaltung einer Qlik Sense-Site

Mit Qlik Management Console können Sie Qlik Sense-Sites einfach und intuitiv konfigurieren, verwalten und überwachen. Sie ermöglicht zudem die Lizenzverwaltung, die Festlegung von Zugriffs- und Sicherheitsregeln, die Konfiguration von Knoten und Datenquellenverbindungen sowie die Synchronisierung von Inhalten und Benutzern übergreifend über viele andere Aktivitäten und Ressourcen.

1.5 Erweiterung von Qlik Sense und individuelle Anpassung

Qlik Sense bietet Ihnen flexible APIs und SDKs zur Entwicklung Ihrer eigenen Erweiterungen und Anpassung und Integration in Qlik Sense für verschiedene Zielsetzungen, unter anderem:

Erstellung von Erweiterungen und Mashups

Hier können Sie für die Web-Entwicklung JavaScript zur Erstellung von Erweiterungen nutzen, die als benutzerdefinierte Visualisierungen in Qlik Sense-Apps dienen, oder Mashup-APIs verwenden, um Websites mit Qlik Sense-Inhalten zu erstellen.

Erstellung von Clients

Sie können in .NET Clients erstellen und Qlik Sense-Objekte in Ihre eigenen Anwendungen einbetten. Außerdem lassen sich native Clients in einer beliebigen Programmiersprache erstellen, die sich für die WebSocket-Kommunikation mithilfe des Qlik Sense-Client-Protokolls eignet.

Erstellung von Servertools

Service- und Benutzerverzeichnis-APIs ermöglichen Ihnen die Erstellung Ihres eigenen Tools zur Administration und Verwaltung von Qlik Sense-Sites.

Herstellung von Verbindungen zu anderen Datenquellen

Erstellen Sie Qlik Sense-Konnektoren zum Abrufen von Daten aus benutzerdefinierten Datenquellen.

2 Skript-Syntax – Überblick

2.1 Einführung in Skriptsyntax

In einem Skript sind die Datenquelle und die zu ladenden Tabellen und Felder definiert. Auch die Zugriffskontrolle ist durch das Skript geregelt. Ein Skript ist eine Aneinanderreihung von Skriptbefehlen, die nacheinander ausgeführt werden.

Die Syntax der Qlik Sense Befehlszeile und die Skriptsyntax entsprechen in ihrer Notation dem Backus-Naur-Formalismus, auch BNF-Code genannt.

Die ersten Codezeilen werden automatisch generiert, wenn eine neue Qlik Sense-Datei erstellt wird. Die Standardwerte dieser Variablen zur Interpretation von Zahlen werden anhand der Regionaleinstellungen des Betriebssystems festgelegt.

Das Skript ist eine Aneinanderreihung von Skriptbefehlen, die nacheinander ausgeführt werden. Jeder Skriptbefehl muss mit einem Semikolon "**;**" enden.

Sie können in den **LOAD**-Befehlen Formeln und Funktionen verwenden, um die geladenen Daten umzuwandeln.

Für Tabellen, in denen Kommas, Tabulatoren oder Semikolons als Trennzeichen verwendet werden, kann ein **LOAD**-Befehl verwendet werden. Standardmäßig lädt ein **LOAD**-Befehl alle Felder der Datei.

Der Zugriff auf allgemeine Datenbanken kann über ODBC- oder OLE DB-Datenbankkonnektoren erfolgen. Dabei werden die üblichen SQL-Befehle verwendet. Die SQL-Syntax muss abhängig vom ODBC-Treiber variiert werden.

Mit benutzerdefinierten Konnektoren können Sie außerdem auf weitere Datenquellen zugreifen.

2.2 Was ist der Backus-Naur-Formalismus?

Die Befehlszeilensyntax und die Skriptsyntax von Qlik Sense entsprechen in ihrer Notation dem Backus-Naur-Formalismus, auch als BNF-Code bekannt.

Die folgende Tabelle enthält eine Liste der Symbole, die im BNF-Code verwendet werden, mit einer Beschreibung, wie sie interpretiert werden:

Symbole

Symbol	Beschreibung
|	Logisches OR, d. h. das Symbol rechts oder links des Strichs kann benutzt werden.
( )	Runde Klammern definieren eine Reihenfolge und strukturieren die BNF-Syntax.
[ ]	Eckige Klammern kennzeichnen optionale Elemente.
{ }	Geschweifte Klammern kennzeichnen Elemente, die gar nicht, einmal oder mehrmals vorkommen können.

Symbol	Beschreibung
Symbol	Eine beliebig lange syntaktische Kategorie, die weiter in andere Symbole unterteilt werden kann. Zum Beispiel Zusammensetzungen der oben genannten Elemente, anderen beliebig langen Symbolen, Strings usw.
::=	Kennzeichnet den Beginn der Definition eines Symbols.
LADEN	Ein direktes Symbol, bestehend aus einem String. Es sollte exakt in der angegebenen Form in das Skript übernommen werden.

Alle direkten Symbole sind **bold face** gedruckt. Zum Beispiel wird "(" als runde Klammern, die eine Reihenfolge definieren, interpretiert, aber "**(**" sollte als Zeichen interpretiert werden, das im Skript gedruckt werden sollte.

Beispiel:

Die Beschreibung des alias-Befehls lautet folgendermaßen:

alias fieldname **as** aliasname { , fieldname **as** aliasname}

Diese Notation ist wie folgt zu verstehen: Der Textstring „alias“, gefolgt von einem beliebigen Feldnamen, gefolgt von dem Textstring „as“, gefolgt von einem beliebigen Aliasnamen. Anschließend können in beliebiger Anzahl Ergänzungen des Typs fieldname as alias angefügt werden, die durch Kommas voneinander getrennt werden.

Folgende Befehle sind demnach korrekt:

```
alias a as first;

alias a as first, b as second;

alias a as first, b as second, c as third;
```

Folgende Befehle dagegen sind nicht korrekt:

```
alias a as first b as second;

alias a as first { , b as second };
```

3 Skriptbefehle

Das Qlik Sense-Skript besteht aus einer Reihe von Befehlen. Ein Befehl kann ein regulärer Skriptbefehl oder Steuerungsbefehl sein. Manchen Befehlen kann ein Zusatz voran- oder nachgestellt werden.

Gewöhnliche Befehle dienen dazu, Daten einzulesen und diese Daten zu strukturieren oder zu verändern. Sie können sich über mehrere Zeilen erstrecken und müssen stets mit einem Semikolon enden , ";".

Steuerungsbefehle steuern die Ausführung des Skripts. Sie dürfen nicht über eine Zeile hinausgehen und werden durch ein Semikolon oder eine Zeilenschaltung beendet.

Eine Ergänzung durch Zusätze ist nur für gewöhnliche Befehle, nicht aber für Steuerungsbefehle möglich. Lediglich die Zusätze **when** und **unless** können an bestimmte Steuerungsbefehle angehängt werden.

Im nächsten Abschnitt finden Sie eine alphabetische Übersicht aller Skriptbefehle, Steuerungsbefehle und Zusätze.

Sämtliche Skriptbefehle können in Groß- oder Kleinbuchstaben oder einer Kombination aus beiden eingegeben werden. Bei Feld- und Variablennamen, die im Skript vorkommen, wird Groß- und Kleinschreibung jedoch unterschieden.

3.1 Steuerungsbefehle im Skript

Das Qlik Sense-Skript besteht aus einer Reihe von Befehlen. Ein Befehl kann ein regulärer Skriptbefehl oder Steuerungsbefehl sein.

Steuerungsbefehle steuern die Ausführung des Skripts. Sie dürfen nicht über eine Zeile hinausgehen und werden durch ein Semikolon oder eine Zeilenschaltung beendet.

Steuerungsbefehle werden im Allgemeinen nicht durch Zusätze ergänzt. Nur in wenigen speziellen Fällen können Steuerungsbefehle durch **when** oder **unless** ergänzt werden.

Sämtliche Skriptbefehle können in Groß- oder Kleinbuchstaben oder einer Kombination aus beiden eingegeben werden.

Steuerungsbefehle im Skript – Übersicht

Jede Funktion wird nach der Übersicht genauer beschrieben. Sie können auch auf den Funktionsnamen in der Syntax klicken, um direkt auf die Details zu der spezifischen Funktion zuzugreifen.

Call

Der Steuerungsbefehl **call** ruft eine Subroutine auf, die vorher durch einen **sub**-Befehl definiert werden muss.

```
Call name ( [ paramlist ])
```

Do..loop

Der Steuerungsbefehl **do..loop** definiert eine Skriptiteration, die einen bzw. mehrere Befehle ausführt, bis eine logische Bedingung erfüllt ist.

```
Do..loop [ ( while | until ) condition ] [statements]
[exit do [ ( when | unless ) condition ] [statements]
loop [ ( while | until ) condition ]
```

Exit script

Dieser Steuerungsbefehl beendet die Ausführung des Skripts. Er kann an beliebiger Stelle des Skripts stehen.

```
Exit script[ (when | unless) condition ]
```

For each ..next

Der Steuerungsbefehl **for each..next** definiert eine Skriptiteration, die für jeden Wert in einer kommagetrennten Liste einen oder mehrere Befehle ausführt. Für jeden Wert der Liste werden die Befehle zwischen **for** und **next** einmal ausgeführt.

```
For each..next var in list

[statements]

[exit for [ ( when | unless ) condition ]

[statements]

next [var]
```

For..next

Der Steuerungsbefehl **for..next** ist eine Skriptiteration mit einem Zähler. Für jeden Zähler innerhalb der festgelegten Grenzen werden die Befehle innerhalb der Schleife, die durch **for**und **next** eingeschlossen sind, jeweils einmal ausgeführt.

```
For..next counter = expr1 to expr2 [ stepexpr3 ]

[statements]

[exit for [ ( when | unless ) condition ]

[statements]

Next [counter]
```

If..then

Der Steuerungsbefehl **if..then** ist eine Skriptauswahl, mit der die Ausführung des Skripts gezwungen wird, abhängig von einer oder mehreren logischen Bedingungen unterschiedlichen Pfaden zu folgen.

*Da **if..then** zu den Steuerungsbefehlen gehört und mit einem Semikolon oder einer Zeilenschaltung abschließt, dürfen sich die einzelnen Befehlssequenzen (**if..then**, **elseif..then**, **else** und **end if**) nicht über mehrere Zeilen erstrecken.*

```
If..then..elseif..else..end if condition then
```

```
  [ statements ]

{ elseif condition then

  [ statements ] }

[ else

  [ statements ] ]

end if
```

Sub

Der Steuerungsbefehl **sub..end sub** definiert eine Subroutine, die zu einem späteren Zeitpunkt durch den Befehl **call** aufgerufen werden kann.

```
Sub..end sub name [ ( paramlist )] statements end sub
```

Switch

Der Steuerungsbefehl **switch** ist eine Skriptauswahl, mit der die Ausführung des Skripts gezwungen wird, abhängig vom Wert einer Formel unterschiedlichen Pfaden zu folgen.

```
Switch..case..default..end switch expression {case valuelist [ statements ]}
[default statements] end switch
```

Call

Der Steuerungsbefehl **call** ruft eine Subroutine auf, die vorher durch einen **sub**-Befehl definiert werden muss.

Syntax:

```
Call name ( [ paramlist ])
```

Argumente:

Argumente

Argument	Beschreibung
name	Der Name der Subroutine.
paramlist	Eine durch Komma getrennte Liste der Parameter, die an die Subroutine übergeben werden. Jeder Eintrag dieser Liste kann ein Feldname, eine Variable oder eine beliebige Formel sein.

Die Subroutine, die von einem **call**-Befehl aus aufgerufen wird, muss bereits an einer vorherigen Stelle im Skript durch ein **sub** definiert sein.

Die Parameter werden dabei in die Subroutine kopiert und, sofern es sich beim Parameter im **call**-Befehl um eine Variable und nicht um eine Formel handelt, beim Beenden der Subroutine auch wieder zurückkopiert.

Beschränkungen:

- Da **call** zu den Steuerungsbefehlen gehört und daher mit einem Semikolon oder einer Zeilenschaltung abschließt, darf sich der Befehl nicht über mehrere Zeilen erstrecken.
- Wenn Sie eine Unterroutine mit `Sub..end sub` innerhalb eines Steuerungsbefehls definieren, beispielsweise `if..then`, können Sie nur die Unterroutine innerhalb des gleichen Steuerungsbefehls aufrufen.

Beispiel:

Dieses Beispiel lädt eine Liste aller Qlik-relevanten Dateien in einem Ordner und seinen Unterordnern. Die Dateiinformationen werden in einer Tabelle gespeichert. Es wird angenommen, dass Sie eine Datenverbindung mit dem Namen Apps zu diesem Ordner erstellt haben.

Die DoDir-Subroutine wird mit der Referenz auf den Ordner 'lib://Apps' als Parameter aufgerufen. In der Subroutine befindet sich eine Rekursion `Call DoDir (Dir)`. Diese sorgt dafür, dass die Funktion in den Unterordnern rekursiv nach Dateien sucht.

```
sub DoDir (Root)
   For Each Ext in 'qvw', 'qvo', 'qvs', 'qvt', 'qvd', 'qvc', 'qvf'
      For Each File in filelist (Root&'\*.' &Ext)
          LOAD
             '$(File)' as Name,
             FileSize( '$(File)' ) as Size,
             FileTime( '$(File)' ) as FileTime
          autogenerate 1;
      Next File
   Next Ext
   For Each Dir in dirlist (Root&'\*' )
      Call DoDir (Dir)
   Next Dir
End Sub

Call DoDir ('lib://Apps')
```

Do..loop

Der Steuerungsbefehl **do..loop** definiert eine Skriptiteration, die einen bzw. mehrere Befehle ausführt, bis eine logische Bedingung erfüllt ist.

Syntax:

```
Do [ ( while | until ) condition ] [statements]
[exit do [ ( when | unless ) condition ] [statements]
loop[ ( while | until ) condition ]
```

*Da **do..loop** zu den Steuerungsbefehlen gehört und mit einem Semikolon oder einer Zeilenschaltung abschließt, dürfen sich die einzelnen Befehlssequenzen (**do**, **exit do** und **loop**) nicht über mehrere Zeilen erstrecken.*

Argumente:

Argumente

Argument	Beschreibung
condition	Eine logische Formel, die True oder False ergibt.
statements	Jede Gruppe von einem oder mehreren Qlik Sense-Skriptbefehlen.
while / until	Die **while**- oder **until**-Bedingungen dürfen nur einmal im **do..loop**-Befehl auftreten, d. h. entweder nach dem **do** oder nach dem **loop**. Die logische Prüfung der Bedingung findet für jeden Durchlauf der Schleife jeweils nur einmal statt, auch wenn die Bedingung nochmals in der Schleife erscheint.
exit do	Steht in der Schleife eine **exit do**-Bedingung, wird die Ausführung des Skripts beim ersten Befehl nach der Schleife, d. h. nach der Zeile mit dem abschließenden **loop**, fortgesetzt. Auf **exit do** kann verzichtet werden, wenn stattdessen **when** oder **unless** verwendet wird.

Beispiel:

```
// LOAD files file1.csv..file9.csv

Set a=1;

Do while a<10

LOAD * from file$(a).csv;

Let a=a+1;

Loop
```

End

Das Skriptschlüsselwort **End** wird zum Abschließen der Bedingungen **If**, **Sub** und **Switch** verwendet.

Exit

Das Skriptschlüsselwort **Exit** ist Teil des Befehls **Exit Script**, kann aber auch zum Beenden der Bedingungen **Do**, **For** oder **Sub** verwendet werden.

Exit script

Dieser Steuerungsbefehl beendet die Ausführung des Skripts. Er kann an beliebiger Stelle des Skripts stehen.

Syntax:

```
Exit Script [ (when | unless) condition ]
```

Da **exit script** zu den Steuerungsbefehlen gehört und daher mit einem Semikolon oder einer Zeilenschaltung abschließt, darf sich der Befehl nicht über mehrere Zeilen erstrecken.

Argumente:

Argumente

Argument	Beschreibung
condition	Eine logische Formel, die True oder False ergibt.
when / unless	Auf den **exit script**-Zusatz kann verzichtet werden, wenn stattdessen **when** oder **unless** verwendet wird.

Beispiele:

```
//Exit script
Exit Script;

//Exit script when a condition is fulfilled
Exit Script when a=1
```

For..next

Der Steuerungsbefehl **for..next** ist eine Skriptiteration mit einem Zähler. Für jeden Zähler innerhalb der festgelegten Grenzen werden die Befehle innerhalb der Schleife, die durch **for**und **next** eingeschlossen sind, jeweils einmal ausgeführt.

Syntax:

```
For counter = expr1 to expr2 [ step expr3 ]

[statements]

[exit for [ ( when | unless ) condition ]

[statements]

Next [counter]
```

Die Formeln *expr1*, *expr2* und *expr3* werden nur beim ersten Durchlauf der Schleife ausgewertet. Der Wert der Zählervariablen kann durch Befehle innerhalb der Schleife geändert werden, dies ist aber nicht empfehlenswert.

Steht in der Schleife eine **exit for**-Bedingung, wird die Ausführung des Skripts beim ersten Befehl nach der Schleife, d. h. nach der Zeile mit dem abschließenden **next**, fortgesetzt. Auf **exit for** kann verzichtet werden, wenn stattdessen **when** oder **unless** verwendet wird.

*Da **for..next** zu den Steuerungsbefehlen gehört und mit einem Semikolon oder einer Zeilenschaltung abschließt, dürfen sich die einzelnen Befehlssequenzen (**for..to..step**, **exit for** und **next**) nicht über mehrere Zeilen erstrecken.*

Argumente:

Argumente

Argument	Beschreibung
counter	Ein Variablenname. Ist *counter* nach **next** festgelegt, muss sie denselben Variablennamen haben, wie den hinter dem zugehörigen **for**.
expr1	Eine Formel, deren Ergebnis den ersten Wert der Variable *counter* ergibt, für welche die Schleife ausgeführt wird.
expr2	Eine Formel, deren Ergebnis den letzten Wert der Variable *counter* ergibt, für welche die Schleife ausgeführt wird.
expr3	Eine Formel, deren Ergebnis den Zuwachs von *counter* ergibt, um den sich der Zähler bei jedem Durchlauf der Schleife erhöht.
condition	Eine logische Formel, die True oder False ergibt.
statements	Jede Gruppe von einem oder mehreren Qlik Sense-Skriptbefehlen.

Example 1: Laden einer Reihe von Dateien

```
// LOAD files file1.csv..file9.csv

for a=1 to 9
      LOAD * from file$(a).csv;
next
```

Example 2: Laden einer zufälligen Zahl von Dateien

In diesem Beispiel werden die Datendateien *x1.csv, x3.csv, x5.csv, x7.csv* und *x9.csv* angenommen. Der Ladevorgang wird an zufällig ausgewählten Punkten mit der Bedingung `if rand( )<0.5 then` gestoppt.

```
for counter=1 to 9 step 2
      set filename=x$(counter).csv;
```

```
	if rand( )<0.5 then
		exit for unless counter=1
	end if

	LOAD a,b from $(filename);
next
```

For each..next

Der Steuerungsbefehl **for each..next** definiert eine Skriptiteration, die für jeden Wert in einer kommagetrennten Liste einen oder mehrere Befehle ausführt. Für jeden Wert der Liste werden die Befehle zwischen **for** und **next** einmal ausgeführt.

Syntax:
Durch eine besondere Syntax ist es möglich, eine Liste mit Verzeichnis- und Dateinamen zu generieren.

```
for each var in list

[statements]

[exit for [ ( when | unless ) condition ]

[statements]

next [var]
```

Argumente:

Argumente

Argument	Beschreibung
var	Ein Skriptvariablen name, der bei jedem Durchlauf der Schleife den jeweils nächsten Wert in der Werteliste annimmt. Ist **var** nach **next** festgelegt, muss sie denselben Variablennamen haben, wie den hinter dem zugehörigen **for each**.

Der Wert von **var** kann durch Befehle innerhalb der Schleife geändert werden, dies ist aber nicht empfehlenswert.

Steht in der Schleife eine **exit for**-Bedingung, wird die Ausführung des Skripts beim ersten Befehl nach der Schleife, d. h. nach der Zeile mit dem abschließenden **next**, fortgesetzt. Auf **exit for** kann verzichtet werden, wenn stattdessen **when** oder **unless** verwendet wird.

*Da **for each..next** zu den Steuerungsbefehlen gehört und mit einem Semikolon oder einer Zeilenschaltung abschließt, dürfen sich die einzelnen Befehlssequenzen (**for each**, **exit for** und **next**) nicht über mehrere Zeilen erstrecken.*

Syntax:

```
list := item { , item }
```

```
item := constant | (expression) | filelist mask | dirlist mask |
fieldvaluelist mask
```

Argumente

Argument	Beschreibung
constant	Beliebige Zahl oder String. Beachten Sie, dass ein direkt im Skript geschriebener String zwischen einfachen Anführungszeichen stehen muss. Ein String ohne einfache Anführungszeichen wird als Variable interpretiert. Anschließend wird der Wert der Variable verwendet. Zahlen müssen nicht zwischen einfachen Anführungszeichen stehen.
expression	Eine beliebige Formel.
mask	Eine Vorgabe für Dateinamen bzw. Ordnernamen, die alle in Dateinamen zugelassenen Zeichen enthalten kann, sowie die Wildcards ***** und **?**. Sie können absolute Dateipfade oder lib://-Pfade verwenden.
condition	Eine logische Formel, die True oder False ergibt.
statements	Jede Gruppe von einem oder mehreren Qlik Sense-Skriptbefehlen.
filelist mask	Durch die Syntax wird eine kommagetrennte Liste aller Dateien im aktuellen Verzeichnis generiert, die der Dateinamenmaske entsprechen. *Dieses Argument unterstützt im Standardmodus nur Bibliotheksverbindungen.*
dirlist mask	Durch die Syntax wird eine kommagetrennte Liste aller Ordner im aktuellen Ordner generiert, die der Ordnernamenmaske entsprechen. *Dieses Argument unterstützt im Standardmodus nur Bibliotheksverbindungen.*
fieldvaluelist mask	Diese Syntax durchläuft die Werte eines Felds, das bereits in Qlik Sense geladen wurde.

*Qlik Konnektoren für Webspeicher-Anbieter und andere DataFiles-Verbindungen unterstützen keine Filtermasken, die Platzhalterzeichen (***** und **?**) enthalten.*

Example 1: Laden einer Liste von Dateien

```
// LOAD the files 1.csv, 3.csv, 7.csv and xyz.csv
for each a in 1,3,7,'xyz'
   LOAD * from file$(a).csv;
next
```

Example 2: Laden einer Dateiliste auf Speichermedium

In diesem Beispiel wird eine Liste aller relevanten Qlik Sense-Dateien in einen Ordner geladen.

```
sub DoDir (Root)
   for each Ext in 'qvw', 'qva', 'qvo', 'qvs', 'qvc', 'qvf', 'qvd'

      for each File in filelist (Root&'/*.' &Ext)

         LOAD
            '$(File)' as Name,
            FileSize( '$(File)' ) as Size,
            FileTime( '$(File)' ) as FileTime
         autogenerate 1;

      next File

   next Ext
   for each Dir in dirlist (Root&'/*' )

      call DoDir (Dir)

   next Dir

end sub

call DoDir ('lib://DataFiles')
```

Example 3: Aggregierung nach den Werten eines Felds

In diesem Beispiel wird die Liste der geladenen Werte von FIELD aggregiert und ein neues Feld – NEWFIELD – erstellt. Für jeden Wert von FIELD werden zwei Datensätze NEWFIELD erstellt.

```
load * inline [
FIELD
one
two
three
];

FOR Each a in FieldValueList('FIELD')
LOAD '$(a)' &'-'&RecNo() as NEWFIELD AutoGenerate 2;
NEXT a
```

Die sich ergebende Tabelle sieht folgendermaßen aus:

Example table

NEWFIELD
one-1
one-2

NEWFIELD
two-1
two-2
three-1
three-2

If..then..elseif..else..end if

Der Steuerungsbefehl **if..then** ist eine Skriptauswahl, mit der die Ausführung des Skripts gezwungen wird, abhängig von einer oder mehreren logischen Bedingungen unterschiedlichen Pfaden zu folgen.

Steuerungsbefehle steuern den Ablauf der Skriptausführung. In einer Diagrammformel verwenden Sie stattdessen die Konditionalfunktion **if**.

Syntax:

```
If  condition then

  [ statements ]

{ elseif condition then

  [ statements ] }

[ else

  [ statements ] ]

end if
```

Da **if..then** zu den Steuerungsbefehlen gehört und mit einem Semikolon oder einer Zeilenschaltung abschließt, dürfen sich die einzelnen Befehlssequenzen (**if..then**, **elseif..then**, **else** und **end if**) nicht über mehrere Zeilen erstrecken.

Argumente:

Argumente

Argument	Beschreibung
condition	Eine logische Formel, die True oder False ergibt.
statements	Jede Gruppe von einem oder mehreren Qlik Sense-Skriptbefehlen.

Example 1:

```
if a=1 then
```

```
      LOAD * from abc.csv;

      SQL SELECT e, f, g from tab1;
end if
```

Example 2:

```
if a=1 then; drop table xyz; end if;
```

Example 3:

```
if x>0 then
      LOAD * from pos.csv;
elseif x<0 then
      LOAD * from neg.csv;
else
      LOAD * from zero.txt;
end if
```

Next

Das Skriptschlüsselwort **Next** wird zum Schließen von zirkulären **For**-Verknüpfungen verwendet.

Sub..end sub

Der Steuerungsbefehl **sub..end sub** definiert eine Subroutine, die zu einem späteren Zeitpunkt durch den Befehl **call** aufgerufen werden kann.

Syntax:

```
Sub name [ ( paramlist )] statements end sub
```

Argumente werden in die Subroutine kopiert. Wenn es sich bei dem entsprechend aufgeführten Parameter im **call**-Befehl um einen Variablennamen handelt, werden sie beim Beenden der Subroutine wieder zurückkopiert.

Wenn in einer Subroutine mehr Parameter definiert sind, als aus dem **call**-Befehl übernommen werden, erhalten die übrigen Parameter den NULL-Wert und können als lokale Variable in der Subroutine verwendet werden.

Argumente:

Argumente

Argument	Beschreibung
name	Der Name der Subroutine.

Argument	Beschreibung
paramlist	Eine kommagetrennte Liste von Variablen, die als Parameter in der Subroutine dienen. Sie können wie jede Variable innerhalb der Subroutine verwendet werden.
statements	Jede Gruppe von einem oder mehreren Qlik Sense-Skriptbefehlen.

Beschränkungen:

- Da **sub** zu den Steuerungsbefehlen gehört und mit einem Semikolon oder einer Zeilenschaltung abschließt, dürfen sich die einzelnen Befehlssequenzen (**sub** und **end sub**) nicht über mehrere Zeilen erstrecken.
- Wenn Sie eine Unterroutine mit `Sub..end sub` innerhalb eines Steuerungsbefehls definieren, beispielsweise `if..then`, können Sie nur die Unterroutine innerhalb des gleichen Steuerungsbefehls aufrufen.

Example 1:

```
Sub INCR (I,J)

I = I + 1

Exit Sub when I < 10

J = J + 1

End Sub

Call INCR (X,Y)
```

Example 2: - Übernahme von Parametern

```
Sub ParTrans (A,B,C)

A=A+1

B=B+1

C=C+1

End Sub

A=1

X=1

C=1

Call ParTrans (A, (X+1)*2)
```

Das Ergebnis des obigen Vorgangs besteht darin, dass A lokal, innerhalb der Subroutine, den Wert 1 erhält, B den Wert 4 und C den Wert NULL.

Beim Abschluss der Subroutine erhält die globale Variable A den Wert 2 (aus der Subroutine zurückkopiert). Der zweite aufgeführte Parameter „(X+1)*2“ wird nicht zurückkopiert, da es sich nicht um eine Variable handelt. Die globale Variable C wird vom Subroutinen-Call nicht beeinflusst.

Switch..case..default..end switch

Der Steuerungsbefehl **switch** ist eine Skriptauswahl, mit der die Ausführung des Skripts gezwungen wird, abhängig vom Wert einer Formel unterschiedlichen Pfaden zu folgen.

Syntax:

```
Switch expression {case valuelist [ statements ]} [default statements] end switch
```

> *Da **switch** zu den Steuerungsbefehlen gehört und mit einem Semikolon oder einer Zeilenschaltung abschließt, dürfen sich die einzelnen Befehlssequenzen (**switch**, **case**, **default** und **end switch**) nicht über mehrere Zeilen erstrecken.*

Argumente:

Argumente

Argument	Beschreibung
expression	Eine beliebige Formel.
valuelist	Eine durch Komma getrennte Liste von Werten, die mit dem Ergebnis der Formel verglichen wird. Die Ausführung des Skripts wird bei den Befehlen fortgesetzt, in deren zugehöriger Werteliste als Erstes eine Übereinstimmung mit dem Ergebnis der Formel festgestellt wird. Die Werte in der Werteliste können beliebige Formeln sein. Wird in keiner der **case**-Bedingungen eine Übereinstimmung festgestellt, werden die auf **default** folgenden Befehle ausgeführt, sofern vorhanden.
statements	Jede Gruppe von einem oder mehreren Qlik Sense-Skriptbefehlen.

Beispiel:

```
Switch I

Case 1

LOAD '$(I): CASE 1' as case autogenerate 1;

Case 2

LOAD '$(I): CASE 2' as case autogenerate 1;

Default
```

```
LOAD '$(I): DEFAULT' as case autogenerate 1;

End Switch
```

To

Das Skriptschlüsselwort **To** wird in verschiedenen Skriptbefehlen verwendet.

3.2 Skript-Zusätze

Eine Ergänzung durch Zusätze ist nur für gewöhnliche Befehle, nicht aber für Steuerungsbefehle möglich. Lediglich die Zusätze **when** und **unless** können an bestimmte Steuerungsbefehle angehängt werden.

Sämtliche Skriptbefehle können in Groß- oder Kleinbuchstaben oder einer Kombination aus beiden eingegeben werden. Bei Feld- und Variablennamen, die im Skript vorkommen, wird Groß- und Kleinschreibung jedoch unterschieden.

Skript-Zusätze – Übersicht

Jede Funktion wird nach der Übersicht genauer beschrieben. Sie können auch auf den Funktionsnamen in der Syntax klicken, um direkt auf die Details zu der spezifischen Funktion zuzugreifen.

Add

Der Zusatz **Add** kann zu jedem **LOAD**- oder **SELECT**-Befehl im Skript hinzugefügt werden, um anzugeben, dass Datensätze zu einer anderen Tabelle hinzugefügt werden sollen. Er gibt auch an, dass dieser Befehl in einem partiellen Ladevorgang ausgeführt werden soll. Der Zusatz **Add** kann auch in einem **Map**-Befehl verwendet werden.

```
Add [only] [Concatenate[(tablename )]] (loadstatement | selectstatement)
```

```
Add [ Only ] mapstatement
```

Buffer

QVD-Dateien können mit Hilfe des **buffer**-Zusatzes im Skript automatisch generiert werden. Dieser Zusatz kann zusammen mit den meisten **LOAD**- und **SELECT**-Befehlen verwendet werden. Er dient dazu, die eingelesenen Daten in einer QVD-Datei zu cachen bzw. zu buffern.

```
Buffer[(option [ , option])] ( loadstatement | selectstatement )
```

```
option::= incremental | stale [after] amount [(days | hours)]
```

Concatenate

Sollen zwei Tabellen zusammengefasst werden, die über unterschiedliche Gruppen von Feldern verfügen, kann die Zusammenfassung zweier Tabellen mit dem Zusatz **Concatenate** erreicht werden.

```
Concatenate[ (tablename ) ] ( loadstatement | selectstatement )
```

Crosstable

Das Ladepräfix **crosstable** wird verwendet, um als „Kreuztabelle“ oder „Pivottabelle“ strukturierte Daten umzuwandeln. Derart strukturierte Daten liegen in der Regel vor, wenn mit Arbeitsblattquellen gearbeitet wird. Die Ausgabe und das Ziel des Ladepräfixes **crosstable** bestehen darin, diese Strukturen in eine entsprechende reguläre spaltenbasierte Tabelle umzuwandeln, da sich diese Struktur meist besser für die Analyse in Qlik Sense eignet.

```
Crosstable (attribute field name, data field name [ , n ] ) ( loadstatement | selectstatement )
```

First

Der Zusatz **First** vor einem **LOAD**- oder **SELECT (SQL)**-Befehl dient dazu, eine festgelegte maximale Anzahl von Datensätzen aus einer Datenquelle zu laden.

```
First n( loadstatement | selectstatement )
```

Generic

Das Ladepräfix **Generic** ermöglicht die Konvertierung von Daten eines Element-Attribut-Wert-Modells (Entity-Attribute-Value, EAV) in eine herkömmliche, normalisierte relationale Tabellenstruktur. EAV-Modelle werden alternativ als „generische Datenmodelle“ oder „offenes Schema“ bezeichnet.

```
Generic ( loadstatement | selectstatement )
```

Hierarchy

Mithilfe des **hierarchy**-Zusatzes wird eine über-/untergeordnete Hierarchie-Tabelle in eine Tabelle umgewandelt, die in einem Qlik Sense-Datenmodell verwendet werden kann. Er kann vor dem Befehl **LOAD** oder **SELECT** eingefügt werden und verwendet anschließend das Ergebnis des Ladebefehls als Eingabe für eine Tabellenumformung.

```
Hierarchy (NodeID, ParentID, NodeName, [ParentName], [PathSource], [PathName], [PathDelimiter], [Depth])(loadstatement | selectstatement)
```

HierarchBelongsTo

Mithilfe dieses Zusatzes wird eine über-/untergeordnete Hierarchie-Tabelle in eine Tabelle umgewandelt, die in einem Qlik Sense-Datenmodell verwendet werden kann. Er kann vor dem Befehl **LOAD** oder **SELECT** eingefügt werden und verwendet anschließend das Ergebnis des Ladebefehls als Eingabe für eine Tabellenumformung.

```
HierarchyBelongsTo (NodeID, ParentID, NodeName, AncestorID, AncestorName, [DepthDiff])(loadstatement | selectstatement)
```

Inner

Dem Präfix **join** und dem Präfix **keep** kann das Präfix **inner** vorangestellt sein.

Vor einem **join**-Befehl bestimmt dieser Zusatz, dass ein Inner Join hergestellt werden soll. Die entstandene Datei enthält somit lediglich Kombinationen von Feldwerten der Datentabellen, wobei die Referenzen zu

externen Werten in beiden Tabellen dargestellt werden. Bei Verwendung vor **keep** bestimmt dieser Zusatz, dass beide Rohdatentabellen auf ihre gemeinsame Schnittmenge reduziert werden sollen, bevor sie in Qlik Sense gespeichert werden.
durch.

```
Inner ( Join | Keep) [ (tablename) ](loadstatement |selectstatement )
```

IntervalMatch
Mithilfe des Zusatzes **IntervalMatch** wird eine Tabelle angelegt, in der die diskreten numerischen Werte mit einem oder mehreren numerischen Intervallen abgeglichen werden und optional die Werte von einem oder mehreren zusätzlichen Schlüsseln.

```
IntervalMatch (matchfield)(loadstatement | selectstatement )
```

```
IntervalMatch (matchfield,keyfield1 [ , keyfield2, ... keyfield5 ] )
(loadstatement | selectstatement )
```

Join
Der **join**-Zusatz kombiniert die geladene Tabelle mit einer bestehenden benannten Tabelle bzw. mit der zuletzt erstellten Datentabelle.

```
[Inner | Outer | Left | Right ] Join [ (tablename ) ]( loadstatement |
selectstatement )
```

Keep
Der Zusatz **keep** weist Ähnlichkeiten mit dem Zusatz **join** auf. Wie der **join**-Zusatz kombiniert er die geladene Tabelle mit einer bestehenden benannten Tabelle oder der zuletzt erstellten Datentabelle, doch statt die geladene Tabelle mit einer bestehenden Tabelle zusammenzuschließen, bewirkt der Zusatz, dass die Tabelle oder beide Tabellen vor dem Speichern in Qlik Sense auf Basis der Schnittmenge der Tabellendaten reduziert werden. Der ausgeführte Vergleich entspricht einer Verknüpfung von Tabellen über alle gemeinsamen Felder, d. h. in gleicher Weise wie bei einer entsprechenden Verknüpfung. Die Tabellen werden jedoch nicht zusammengeschlossen, sondern als zwei Tabellen unter verschiedenen Namen in Qlik Sense gespeichert.

```
(Inner | Left | Right) Keep [(tablename ) ]( loadstatement | selectstatement
)
```

Left
Den Befehlen **Join**und **Keep** kann der Zusatz **left** vorangestellt werden.

Vor einem **join**-Befehl bestimmt dieser Zusatz, dass ein Left Join hergestellt werden soll. Die entstandene Tabelle enthält somit lediglich Kombinationen von Feldwerten der Datentabellen, wobei die Referenzen zu Feldwerten in der ersten Tabelle dargestellt werden. Bei Verwendung vor **keep** bestimmt dieser Zusatz, dass die zweite Rohdatentabelle auf ihre gemeinsame Schnittmenge mit der ersten Tabelle reduziert werden soll, bevor sie in Qlik Sense gespeichert wird.

```
Left ( Join | Keep) [ (tablename) ](loadstatement |selectstatement )
```

Mapping
Mithilfe des Zusatzes **mapping** wird eine Mapping-Tabelle erstellt, die z. B. zum Austauschen der Feldwerte und Feldnamen während der Ausführung des Skripts verwendet werden kann.

```
Mapping  ( loadstatement | selectstatement )
```

Merge
Der Zusatz **Merge** kann zu jedem **LOAD**- oder **SELECT**-Befehl im Skript hinzugefügt werden, um anzugeben, dass die geladene Tabelle mit einer anderen Tabelle zusammengeführt werden soll. Er gibt auch an, dass dieser Befehl in einem partiellen Ladevorgang ausgeführt werden soll.

```
Merge [only] [(SequenceNoField [, SequenceNoVar])] On ListOfKeys [Concatenate
[(TableName)]] (loadstatement | selectstatement)
```

NoConcatenate
Der Zusatz **NoConcatenate** bewirkt, dass zwei geladene Tabellen mit identischen Feldergruppen als zwei separate interne Tabellen behandelt werden, während sie andernfalls automatisch zusammengefasst werden würden.

```
NoConcatenate( loadstatement | selectstatement )
```

Outer
Dem expliziten Präfix **Join** kann das Präfix **Outer** vorangestellt werden, um einen Outer Join festzulegen. In einem Outer Join werden alle Kombinationen zwischen den zwei Tabellen erzeugt. Die entstandene Tabelle enthält somit Kombinationen von Feldwerten der Rohdatentabellen, wobei die verknüpfenden Feldwerte in beiden Tabellen dargestellt werden. Das Schlüsselwort **Outer** ist optional und ist der standardmäßige Typ, der verwendet wird, wenn kein Join-Präfix festgelegt wird.

```
Outer Join [ (tablename) ](loadstatement |selectstatement )
```

Partial reload
Ein vollständiger Ladevorgang beginnt immer mit dem Löschen aller Tabellen im vorhandenen Datenmodell. Dann wird das Ladeskript ausgeführt.

Bei einem *Partielles Laden (page 103)* geschieht dies nicht. Stattdessen werden alle Tabellen im Datenmodell beibehalten und dann nur die Anweisungen **Load** und **Select** mit den Zusätzen **Add**, **Merge** oder **Replace** ausgeführt. Andere Datentabellen sind von dem Befehl nicht betroffen. Das Argument **only** gibt an, dass die Anweisung nur bei partiellen Ladevorgängen ausgeführt und bei vollständigen Ladevorgängen ignoriert werden soll. Die folgende Tabelle bietet eine Zusammenfassung der Anweisungsausführung für partielle und vollständige Ladevorgänge.

Replace
Der Zusatz **Replace** kann zu jedem **LOAD**- oder **SELECT**-Befehl im Skript hinzugefügt werden, um anzugeben, dass die geladene Tabelle eine andere Tabelle ersetzen soll. Er gibt auch an, dass dieser Befehl in einem partiellen Ladevorgang ausgeführt werden soll. Der Zusatz **Replace** kann auch in einem **Map**-Befehl verwendet werden.

```
Replace [only] [Concatenate[(tablename) ]] (loadstatement | selectstatement)
```

```
Replace [only] mapstatement
```

Right
Den Befehlen **Join**und **Keep** kann der Zusatz **right** vorangestellt werden.

Vor einem **join**-Befehl bestimmt dieser Zusatz, dass ein Right Join hergestellt werden soll. Die entstandene Tabelle enthält somit lediglich Kombinationen von Feldwerten der Datentabellen, wobei die Referenzen zu Feldwerten in der zweiten Tabelle dargestellt werden. Bei Verwendung vor **keep** bestimmt dieser Zusatz, dass die erste Rohdatentabelle auf ihre gemeinsame Schnittmenge mit der zweiten Tabelle reduziert werden sollte, bevor sie in Qlik Sense gespeichert wird.

```
Right (Join | Keep) [(tablename)](loadstatement |selectstatement )
```

Sample
Der Zusatz **sample** vor einem **LOAD**- oder **SELECT**-Befehl dient dazu, eine zufällige Stichprobe von Datensätzen aus einer Datenquelle zu laden.

```
Sample p ( loadstatement | selectstatement )
```

Semantic
Tabellen, die Beziehungen zwischen Datensätzen enthalten, können über ein **semantic**-Präfix geladen werden. Das können beispielsweise Selbstreferenzen in einer Tabelle sein, in der ein Datensatz auf einen anderen verweist, wie übergeordnet, gehört zu oder Vorgänger.

```
Semantic ( loadstatement | selectstatement)
```

Unless
Der Zusatz **unless** definiert eine Bedingung für die Evaluierung eines Befehls bzw. eines exit-Befehls. Es kann somit als Kurzform des vollständigen Befehls **if..end if** betrachtet werden.

```
(Unless condition statement | exitstatement Unless condition )
```

When
Der Zusatz **when** definiert eine Bedingung für die Ausführung eines Befehls bzw. eines exit-Befehls. Es kann somit als Kurzform des vollständigen Befehls **if..end if** betrachtet werden.

```
( When condition statement | exitstatement when condition )
```

Add

Der Zusatz **Add** kann zu jedem **LOAD**- oder **SELECT**-Befehl im Skript hinzugefügt werden, um anzugeben, dass Datensätze zu einer anderen Tabelle hinzugefügt werden sollen. Er gibt auch an, dass dieser Befehl in einem partiellen Ladevorgang ausgeführt werden soll. Der Zusatz **Add** kann auch in einem **Map**-Befehl verwendet werden.

Damit der partielle Ladevorgang korrekt funktioniert, muss die App mit Daten geöffnet werden, bevor der partielle Ladevorgang ausgelöst wird.

Führen Sie einen partiellen Ladevorgang durch, indem Sie die Schaltfläche **Laden** nutzen. Sie können auch das Qlik Engine JSON API verwenden.

Syntax:

```
Add [only] [Concatenate[(tablename)]] (loadstatement | selectstatement)
```

```
Add [only] mapstatement
```

Während eines normalen (nicht partiellen) Ladevorgangs funktioniert die Konstruktion **Add LOAD** wie ein normaler **LOAD**-Befehl. Datensätze werden generiert und in einer Tabelle gespeichert.

Wenn der Zusatz **Concatenate** verwendet wird oder wenn eine Tabelle mit dem gleichen Satz Felder vorhanden ist, werden die Datensätze an die relevante vorhandene Tabelle angehängt. Andernfalls erstellt die Konstruktion **Add LOAD** eine neue Tabelle.

Ein partieller Ladevorgang hat die gleiche Wirkung. Der einzige Unterschied besteht darin, dass mit der Konstruktion **Add LOAD** nie eine neue Tabelle erstellt wird. Es ist immer eine relevante Tabelle aus der vorherigen Skriptausführung vorhanden, an die die Datensätze angehängt werden müssen.

Dabei wird nicht geprüft, ob die Datensätze bereits vorhanden sind. Um Duplikate zu vermeiden, wird der **Add**-Zusatz häufig in Verbindung mit einem distinct-Qualifizierer oder einem where-Befehl benutzt.

Durch den **Add Map...Using**-Befehl wird auch bei der partiellen Ausführung des Skripts ein Mapping durchgeführt.

Argumente:

Argumente

Argument	Beschreibung
only	Ein optionaler Qualifizierer, der bewirkt, dass der Befehl nur bei der partiellen Ausführung des Skripts berücksichtigt wird. Bei normalen (nicht partiellen) Ladevorgängen sollte er ignoriert werden.

Beispiele und Ergebnisse:

Beispiel	Ergebnis

`Tab1:`
 	
`LOAD Name, Number FROM Persons.csv;`
 	
`Add LOAD Name, Number FROM newPersons.csv;`
	Bei der normalen Ausführung des Skripts werden die Daten aus der Datei *Persons.csv* geladen und in der Qlik Sense-Tabelle Tab1 gespeichert. Die Werte aus der Datei *NewPersons.csv* werden in derselben Qlik Sense-Tabelle zusammengefasst.
	
	Bei der partiellen Ausführung des Skripts werden die Daten aus *NewPersons.csv* geladen und in der Qlik Sense-Tabelle Tab1 angehängt. Dabei wird nicht auf Dubletten geprüft.

Beispiel	Ergebnis
Tab1: SQL SELECT Name, Number FROM Persons.csv; Add LOAD Name, Number FROM NewPersons.csv where not exists (Name);	Dabei wird mit überprüft, ob Name in den zuvor geladenen Tabellendaten bereits existiert. Bei der normalen Ausführung des Skripts werden die Daten aus der Datei *Persons.csv* geladen und in der Qlik Sense-Tabelle Tab1 gespeichert. Die Werte aus der Datei *NewPersons.csv* werden in derselben Qlik Sense-Tabelle zusammengefasst. Bei der partiellen Ausführung des Skripts werden die Daten aus *NewPersons.csv* geladen und in der Qlik Sense-Tabelle Tab1 angehängt. Dabei wird mit geprüft, ob Name in den zuvor geladenen Tabellendaten bereits existiert.
Tab1: LOAD Name, Number FROM Persons.csv; Add Only LOAD Name, Number FROM NewPersons.csv where not exists(Name);	Bei der normalen Ausführung des Skripts werden die Daten aus der Datei *Persons.csv* geladen und in der Qlik Sense-Tabelle Tab1 gespeichert. Der Befehl zum Laden von *NewPersons.csv* bleibt unberücksichtigt. Bei der partiellen Ausführung des Skripts werden die Daten aus *NewPersons.csv* geladen und in der Qlik Sense-Tabelle Tab1 angehängt. Dabei wird mit geprüft, ob Name in den zuvor geladenen Tabellendaten bereits existiert.

Buffer

QVD-Dateien können mit Hilfe des **buffer**-Zusatzes im Skript automatisch generiert werden. Dieser Zusatz kann zusammen mit den meisten **LOAD**- und **SELECT**-Befehlen verwendet werden. Er dient dazu, die eingelesenen Daten in einer QVD-Datei zu cachen bzw. zu buffern.

Syntax:

```
Buffer [(option [ , option])] ( loadstatement | selectstatement )
option::= incremental | stale [after] amount [(days | hours)]
```

Ist keine Option definiert, wird die bei der ersten Skriptausführung generierte QVD-Datei auf unbestimmte Zeit weiterverwendet.

Die Pufferdatei ist im Unterordner *Buffers* gespeichert, in der Regel unter *C:\ProgramData\Qlik\Sense\Engine\Buffers* (Serverinstallation) oder *C:\Users\{user}\Documents\Qlik\Sense\Buffers* (Qlik Sense Desktop).

Der Name der QVD-Datei ist ein generierter Name, ein hexadezimaler 160 bit-String, der den nachfolgenden **LOAD**- oder **SELECT**-Befehl und andere Informationen in abgekürzter Form enthält. Wird der nachfolgende QVD- oder **LOAD**-Befehl verändert, wird der **SELECT**-Puffer ungültig.

Normalerweise löscht das Programm automatisch QVD-Puffer, auf die während der Ausführung eines vollständigen Skripts in der App, die ihn erstellt hat, nicht mehr zugegriffen wird. Auch beim Löschen der App werden die Buffer automatisch mitgelöscht.

Argumente:

Argumente

Argument	Beschreibung
incremental	Durch die Option incremental wird nur ein Teil der Daten aus der Originaldatenquelle gelesen. Die bisherige Größe der Quelldatei ist im XML-Header der QVD-Datei gespeichert. Dies ist insbesondere für log-Dateien nützlich. Alle bereits in der QVD-Datei vorhandenen Datensätze werden von dort eingelesen, alle neuen Datensätze aus der Originaldatenquelle. Am Ende dieses Vorgangs wird eine aktualisierte Version in der QVD-Datei gespeichert. Die Option incremental kann nur mit **LOAD**-Befehlen und Textdateien verwendet werden. Inkrementelles Laden kann nicht verwendet werden, wenn alte Daten geändert oder gelöscht werden.
stale [after] amount [(days | hours)]	amount ist eine Zahl, die den Zeitraum definiert. Es dürfen Dezimalzahlen verwendet werden. Ist keine Einheit angegeben, nimmt das Programm Tage an. Die Option stale after ist für Datenbanken vorgesehen, deren Originaldaten keinen Zeitstempel enthalten. Stattdessen definiert man, wie alt der QVD-Schnappschuss maximal sein darf, um anstelle der Originaldatenquelle eingelesen zu werden. Eine Bedingung stale after gibt einfach einen Zeitraum für das Erstellungsdatum des QVD-Buffers an. Danach ist er nicht mehr gültig. Vor dem Zeitrahmen wird der QVD-Buffer als Quelle für Daten verwendet, und danach gilt die ursprüngliche Datenquelle. In diesem Fall wird eine aktualisierte Version der QVD-Datei gespeichert, und die Zeitspanne beginnt von Neuem.

Beschränkungen:

Im Gebrauch des buffer-Zusatzes bestehen einige Einschränkungen. Die wichtigste besteht darin, dass der Zusatz nur für **LOAD**- oder **SELECT**-Befehle benutzt werden darf, die Daten aus einer Datei lesen.

Example 1:

```
Buffer SELECT * from MyTable;
```

Example 2:

```
Buffer (stale after 7 days) SELECT * from MyTable;
```

Example 3:

```
Buffer (incremental) LOAD * from MyLog.log;
```

Concatenate

`Concatenate` ist ein Skriptladepräfix, mit dem ein Datensatz an eine bereits vorhandene In-Memory-Tabelle angehängt werden kann. Es wird häufig verwendet, um verschiedene Sätze transaktionaler Daten an eine einzelne zentrale Fakttabelle anzuhängen, oder um gemeinsame Referenzdatensätze eines bestimmten Typs zu erstellen, die sich aus mehreren Quellen zusammensetzen. Seine Funktionsweise gleicht einem SQL UNION-Operator.

Die Ergebnistabelle eines `concatenate`-Vorgangs enthält den Originaldatensatz mit den neuen Datenzeilen, die unten an die Tabelle angehängt wurden. In der Quell- und in der Zieltabelle können unterschiedliche Felder vorhanden sein. Wenn Felder unterschiedlich sind, wird die Ergebnistabelle ausgeweitet und stellt die kombinierten Ergebnisse aller Felder dar, die sowohl in der Quelltabelle als auch in der Zieltabelle vorhanden sind.

Syntax:

```
Concatenate[ (tablename ) ] ( loadstatement | selectstatement )
```

Argumente

Argument	Beschreibung
`tablename`	Der Name einer vorhandenen Tabelle. Die benannte Tabelle wird das Ziel des `Concatenate`-Vorgangs, und alle geladenen Datensätze werden an diese Tabelle angehängt. Wenn der Parameter `tablename` nicht verwendet wird, ist die Zieltabelle die letzte geladene Tabelle vor diesem Befehl.
`loadstatement/selectstatement`	Das Argument `loadstatement/selectstatement`, das auf das Argument `tablename` folgt, wird mit der angegebenen Tabelle zusammengefasst.

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Funktionsbeispiel

Beispiel	Ergebnis
`Concatenate (Transactions) Load …. ;`	Die in der load-Anweisung unter dem Präfix `Concatenate` geladenen Daten werden an die vorhandene In-Memory-Tabelle namens `Transactions` angehängt (wobei angenommen wird, dass eine Tabelle mit dem Namen `Transactions` zuvor in das Ladeskript geladen wurde).

Beispiel 1 – Anhängen von mehreren Datensets an eine Zieltabelle mit dem Ladepräfix „Concatenate“

Ladeskript und Ergebnisse

Übersicht

In diesem Beispiel laden Sie zwei Skripts in Reihenfolge.

- Das erste Ladeskript enthält einen anfänglichen Datensatz mit Datumswerten und Beträgen, der an eine Tabelle namens `Transactions` gesendet wird.
- Das zweite Ladeskript enthält:
    - Einen zweiten Datensatz, der an den anfänglichen Datensatz mit dem Präfix `Concatenate` angehängt wird. Dieser Datensatz enthält ein weiteres Feld `type`, das nicht im anfänglichen Datensatz enthalten war.
    - Das Präfix `Concatenate`

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Erstes Ladeskript

```
Transactions:
Load * Inline [

id, date, amount
3750, 08/30/2018, 23.56
3751, 09/07/2018, 556.31
3752, 09/16/2018, 5.75
3753, 09/22/2018, 125.00
3754, 09/22/2018, 484.21
3756, 09/22/2018, 59.18
3757, 09/23/2018, 177.42
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- id
- date
- amount

Ergebnistabelle des ersten Ladeskripts

id	date	amount
3750	08/30/2018	23.56
3751	09/07/2018	556.31
3752	09/16/2018	5.75
3753	09/22/2018	125.00
3754	09/22/2018	484.21
3756	09/22/2018	59.18
3757	09/23/2018	177.42

Die Tabelle zeigt den anfänglichen Datensatz.

Zweites Ladeskript

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten ein.

```
Concatenate(Transactions)
Load * Inline [
id, date, amount, type
3758, 10/01/2018, 164.27, Internal
3759, 10/03/2018, 384.00, External
3760, 10/06/2018, 25.82, Internal
3761, 10/09/2018, 312.00, Internal
3762, 10/15/2018, 4.56, Internal
3763, 10/16/2018, 90.24, Internal
3764, 10/18/2018, 19.32, External
];
```

Ergebnisse

Laden Sie die Daten und wechseln Sie zum Arbeitsblatt. Erstellen Sie dieses Feld als Dimension:

- type

Ergebnistabelle des zweiten Ladeskripts

id	date	amount	type
3750	08/30/2018	23.56	-
3751	09/07/2018	556.31	-
3752	09/16/2018	5.75	-

id	date	amount	type
3753	09/22/2018	125.00	-
3754	09/22/2018	484.21	-
3756	09/22/2018	59.18	-
3757	09/23/2018	177.42	-
3758	10/01/2018	164.27	Intern
3759	10/03/2018	384.00	Extern
3760	10/06/2018	25.82	Intern
3761	10/09/2018	312.00	Intern
3762	10/15/2018	4.56	Intern
3763	10/16/2018	90.24	Intern
3764	10/18/2018	19.32	Extern

Beachten Sie die Nullwerte im Feld `type` für die ersten sieben geladenen Datensätze, wo `type` nicht definiert wurde.

Beispiel 2 – Anhängen von mehreren Datensets an eine Zieltabelle mit impliziter Zusammenfassung

Ladeskript und Ergebnisse

Übersicht

Ein typischer Anwendungsfall für das implizite Anhängen von Daten liegt vor, wenn Sie mehrere Dateien mit identisch strukturierten Daten laden und diese alle an eine Zieltabelle anhängen möchten.

Beispielsweise können `wildcards` in Dateinamen mit einer Syntax wie der folgenden verwendet werden:

```
myTable:
Load * from [myFile_*.qvd] (qvd);
```

oder in Schleifen mit Konstrukten wie:

```
for each file in filelist('myFile_*.qvd')

myTable:
Load * from [$(file)] (qvd);

next file
```

Die implizite Zusammenfassung erfolgt zwischen zwei beliebigen Tabellen, die mit identisch benannten Feldern geladen werden, selbst wenn sie im Skript nicht nacheinander definiert sind. Dies kann dazu führen, dass Daten unabsichtlich an Tabellen angehängt werden. Wenn Sie nicht möchten, dass eine sekundäre Tabelle mit identischen Feldern auf diese Weise angehängt wird, verwenden Sie das Ladepräfix `NoConcatenate`. Das Umbenennen der Tabelle mit einem Tag für alternativen Tabellennamen ist nicht ausreichend, um die implizite Zusammenfassung zu verhindern. Weitere Informationen finden Sie unter NoConcatenate (page 93).

In diesem Beispiel laden Sie zwei Skripts in Reihenfolge.

- Das erste Ladeskript enthält einen anfänglichen Datensatz mit vier Feldern, der an eine Tabelle namens `Transactions` gesendet wird.
- Das zweite Ladeskript enthält einen Datensatz mit den gleichen Feldern wie der erste Datensatz.

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Erstes Ladeskript

```
Transactions:
Load * Inline [
id, date, amount, type
3758, 10/01/2018, 164.27, Internal
3759, 10/03/2018, 384.00, External
3760, 10/06/2018, 25.82, Internal
3761, 10/09/2018, 312.00, Internal
3762, 10/15/2018, 4.56, Internal
3763, 10/16/2018, 90.24, Internal
3764, 10/18/2018, 19.32, External
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `id`
- `date`
- `amount`
- `type`

Ergebnistabelle des ersten Ladeskripts

id	date	type	amount
3758	10/01/2018	Intern	164.27
3759	10/03/2018	Extern	384.00
3760	10/06/2018	Intern	25.82

id	date	type	amount
3761	10/09/2018	Intern	312.00
3762	10/15/2018	Intern	4.56
3763	10/16/2018	Intern	90.24
3764	10/18/2018	Extern	19.32

Die Tabelle zeigt den anfänglichen Datensatz.

Zweites Ladeskript

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten ein.

```
Load * Inline [
id, date, amount, type
3765, 11/03/2018, 129.40, Internal
3766, 11/05/2018, 638.50, External
];
```

Ergebnisse

Laden Sie die Daten und wechseln Sie zum Arbeitsblatt.

Ergebnistabelle des zweiten Ladeskripts

id	date	type	amount
3758	10/01/2018	Intern	164.27
3759	10/03/2018	Extern	384.00
3760	10/06/2018	Intern	25.82
3761	10/09/2018	Intern	312.00
3762	10/15/2018	Intern	4.56
3763	10/16/2018	Intern	90.24
3764	10/18/2018	Extern	19.32
3765	11/03/2018	Intern	129.40
3766	11/05/2018	Extern	638.50

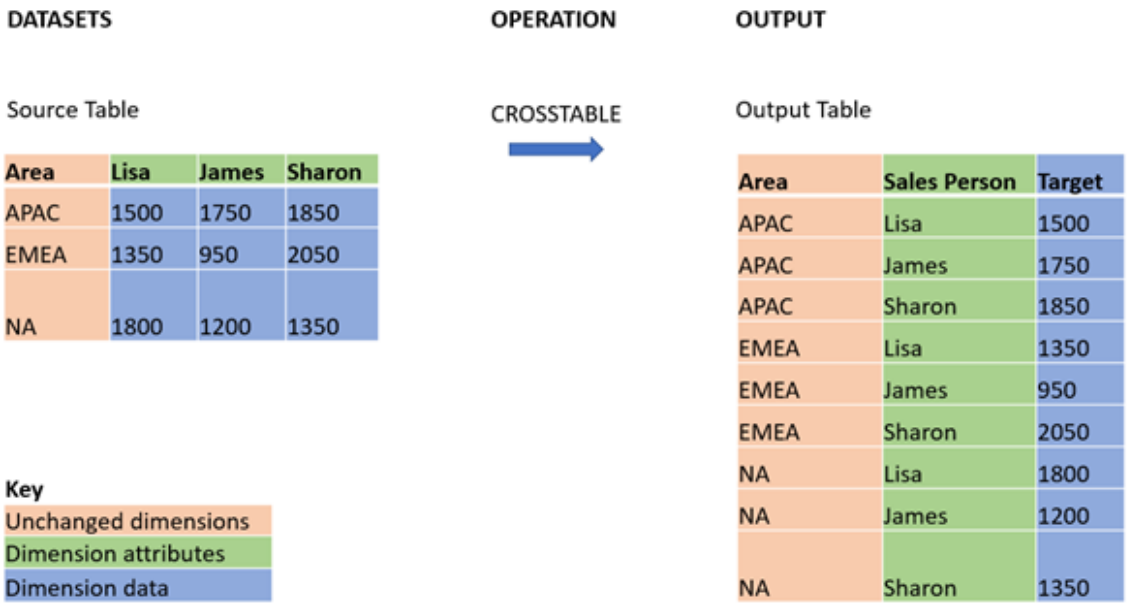
Der zweite Datensatz wurde implizit mit dem anfänglichen Datensatz zusammengefasst, weil sie identische Felder hatten.

Crosstable

Das Ladepräfix **crosstable** wird verwendet, um als „Kreuztabelle“ oder „Pivottabelle“ strukturierte Daten umzuwandeln. Derart strukturierte Daten liegen in der Regel vor, wenn mit Arbeitsblattquellen gearbeitet wird. Die Ausgabe und das Ziel des Ladepräfixes **crosstable** bestehen darin, diese Strukturen in eine entsprechende reguläre spaltenbasierte Tabelle

umzuwandeln, da sich diese Struktur meist besser für die Analyse in Qlik Sense eignet.

Beispiel für als Kreuztabelle strukturierte Daten und die äquivalente Struktur nach einer Kreuztabellenumwandlung.



Syntax:

```
crosstable (attribute field name, data field name [ , n ] ) ( loadstatement | selectstatement )
```

Argumente

Argument	Beschreibung
attribute field name	Der gewünschte Ausgabefeldname, der die horizontal orientierte Dimension beschreibt, die umgewandelt werden soll (die Kopfzeile).
data field name	Der gewünschte Ausgabefeldname, der die horizontal orientierten Daten der Dimension beschreibt, die umgewandelt werden sollen (die Matrix der Datenwerte unter der Kopfzeile).
n	Die Anzahl der Zusatzfelder bzw. unveränderten Dimensionen, die der Tabelle vorangestellt sind und in die generische Form umgewandelt werden sollen. Der Standardwert ist 1.

Diese Skriptfunktion hängt mit folgenden Funktionen zusammen:

Verwandte Funktionen

Funktion	Interaktion
Generic (page 59)	Ein Umwandlungs-Ladezusatz, der einen Datensatz mit Entität-Attribut-Wert-Struktur (EAV) in eine reguläre relationale Tabellenstruktur umwandelt und jedes gefundene Attribut als ein eigenes neues Datenfeld bzw. neue Datenspalte abtrennt.

Beispiel 1 – Umwandeln von pivotierten Umsatzdaten (einfach)

Ladeskripte und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das erste Ladeskript unten in eine neue Registerkarte ein.

Das erste Ladeskript enthält einen Datensatz, auf den später der Skriptzusatz `crosstable` angewendet wird, wobei der Abschnitt, der `crosstable` anwendet, auskommentiert ist. Das bedeutet, dass die Kommentarsyntax verwendet wurde, um diesen Abschnitt im Ladeskript zu deaktivieren.

Das zweite Ladeskript entspricht dem ersten, aber mit unkommentierter Anwendung von `crosstable` (durch Entfernen der Kommentarsyntax). Die Skripte werden auf diese Weise angezeigt, um den Wert der Skripterstellungsfunktion beim Umwandeln von Daten hervorzuheben.

Erstes Ladeskript (Funktion nicht angewendet)

```
tmpData:
//Crosstable (MonthText, Sales)
Load * inline [
Product, Jan 2021, Feb 2021, Mar 2021, Apr 2021, May 2021, Jun 2021
A, 100, 98, 103, 63, 108, 82
B, 284, 279, 297, 305, 294, 292
C, 50, 53, 50, 54, 49, 51];

//Final:
//Load Product,
//Date(Date#(MonthText,'MMM YYYY'),'MMM YYYY') as Month,
//Sales

//Resident tmpData;

//Drop Table tmpData;
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `Product`
- `Jan 2021`
- `Feb 2021`
- `Mar 2021`
- `Apr 2021`
- `May 2021`
- `Jun 2021`

Ergebnistabelle

Produkt	Jan 2021	Feb 2021	Mär 2021	Apr 2021	Mai 2021	Jun 2021
A	100	98	103	63	108	82
B	284	279	297	305	294	292
C	50	53	50	54	49	51

Dieses Skript ermöglicht die Erstellung einer Kreuztabelle mit einer Spalte für jeden Monat und einer Zeile pro Produkt. Im aktuellen Format ist eine Analyse der Daten schwierig. Es wäre wesentlich praktischer, in einer dreispaltigen Tabelle alle Zahlen in einem Feld und alle Monate in einem anderen zu haben. Im nächsten Abschnitt wird erläutert, wie Sie diese Umwandlung der Kreuztabelle vornehmen.

Zweites Ladeskript (Funktion angewendet)

Heben Sie die Kommentierung des Skripts auf, indem Sie die // entfernen. Das Ladeskript sollte folgendermaßen aussehen:

```
tmpData:
Crosstable (MonthText, Sales)
Load * inline [
Product, Jan 2021, Feb 2021, Mar 2021, Apr 2021, May 2021, Jun 2021
A, 100, 98, 103, 63, 108, 82
B, 284, 279, 297, 305, 294, 292
C, 50, 53, 50, 54, 49, 51];

Final:
Load Product,
Date(Date#(MonthText,'MMM YYYY'),'MMM YYYY') as Month,
Sales

Resident tmpData;

Drop Table tmpData;
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- Product
- Month
- Sales

Ergebnistabelle

Produkt	Monat	Verkauf
A	Jan 2021	100

Produkt	Monat	Verkauf
A	Feb 2021	98
A	Mar 2021	103
A	Apr 2021	63
A	May 2021	108
A	Jun 2021	82
B	Jan 2021	284
B	Feb 2021	279
B	Mar 2021	297
B	Apr 2021	305
B	May 2021	294
B	Jun 2021	292
C	Jan 2021	50
C	Feb 2021	53
C	Mar 2021	50
C	Apr 2021	54
C	May 2021	49
C	Jun 2021	51

Nachdem der Skriptzusatz angewendet wurde, ist die Kreuztabelle in ein Tabellendiagramm mit einer Spalte für `Month` und einer Spalte für `Sales` umgewandelt. Das verbessert die Lesbarkeit der Daten.

Beispiel 2 – Umwandeln von pivotierten Umsatzzieldaten in eine vertikale Tabellenstruktur (mittel)

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz, der in eine Tabelle namens „Targets“ geladen wird.
- Der Ladezusatz `crosstable`, der die pivotierten Vertriebsmitarbeiternamen in ein eigenes Feld mit der Bezeichnung `Sales Person` umwandelt.
- Die zugehörigen Umsatzzieldaten, die in einem Feld namens `Target` strukturiert werden.

Ladeskript

```
SalesTargets:
CROSSTABLE([Sales Person],Target,1)
LOAD
*
INLINE [
Area, Lisa, James, Sharon
APAC, 1500, 1750, 1850
EMEA, 1350, 950, 2050
NA, 1800, 1200, 1350
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `Area`
- `Sales Person`

Fügen Sie folgende Kennzahl hinzu:

```
=Sum(Target)
```

Ergebnistabelle

Fläche	**Vertriebsmitarbeiter**	**=Sum(Target)**
APAC	James	1750
APAC	Lisa	1500
APAC	Sharon	1850
EMEA	James	950
EMEA	Lisa	1350
EMEA	Sharon	2050
–	James	1200
–	Lisa	1800
–	Sharon	1350

Wenn Sie die Anzeige der Daten als pivotierte Eingabetabelle replizieren möchten, können Sie eine entsprechende Pivottabelle in einem Arbeitsblatt erstellen.

Gehen Sie folgendermaßen vor:

1. Kopieren Sie die soeben erstellte Tabelle und fügen Sie sie in ein Arbeitsblatt ein.
2. Ziehen Sie das Diagrammobjekt **Pivottabelle** auf die neu erstellte Tabellenkopie. Wählen Sie **Konvertieren** aus.
3. Klicken Sie auf ✔ **Bearbeitung fertig**.
4. Ziehen Sie das Feld `Sales Person` von der Ablage der vertikalen Spalte zur Ablage der horizontalen Spalte.

Die folgende Tabelle zeigt die Daten in ihrer anfänglichen Tabellenform, so wie sie in Qlik Sense angezeigt werden:

Ursprüngliche Ergebnistabelle, wie in Qlik Sense angezeigt

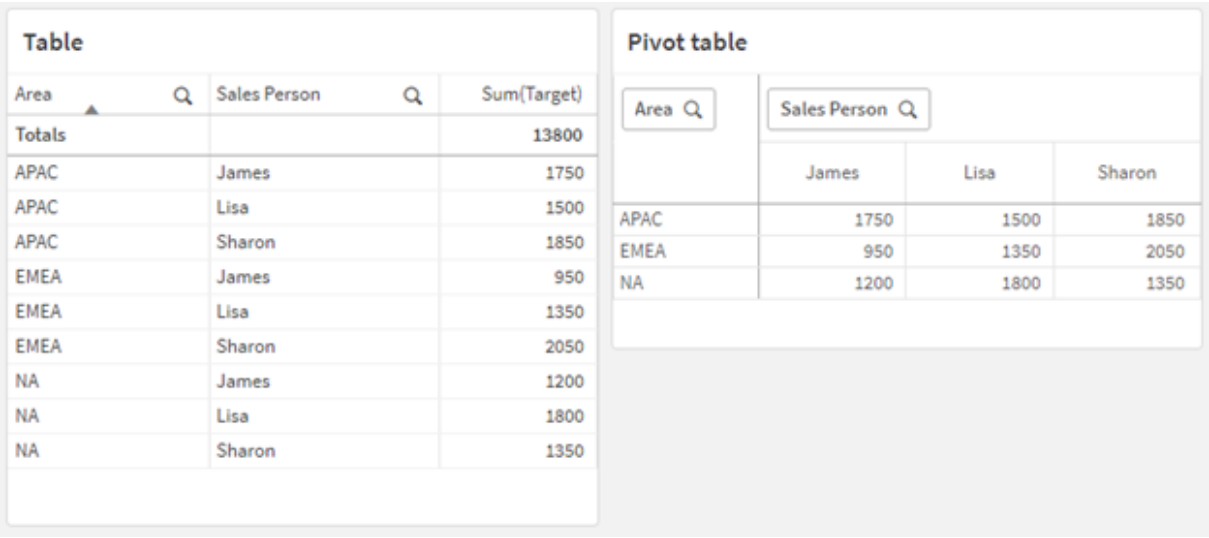
Fläche	**Vertriebsmitarbeiter**	**=Sum(Target)**
Summen	-	13800
APAC	James	1750
APAC	Lisa	1500
APAC	Sharon	1850
EMEA	James	950
EMEA	Lisa	1350
EMEA	Sharon	2050
–	James	1200
–	Lisa	1800
–	Sharon	1350

Die entsprechende Pivottabelle gleicht der folgenden, wobei die Spalte für den Namen der einzelnen Vertriebsmitarbeiter in der längeren Zeile für `Sales Person` enthalten ist:

Entsprechende Pivottabelle, wobei das Feld `Sales Person` horizontal pivotiert ist

Fläche	**James**	**Lisa**	**Sharon**
APAC	1750	1500	1850
EMEA	950	1350	2050
–	1350	1350	1350

Beispiel der als Tabelle angezeigten Daten und einer entsprechenden Pivottabelle, wobei das Feld `Sales Person` horizontal pivotiert ist



Table

Area	Sales Person	Sum(Target)
Totals		13800
APAC	James	1750
APAC	Lisa	1500
APAC	Sharon	1850
EMEA	James	950
EMEA	Lisa	1350
EMEA	Sharon	2050
NA	James	1200
NA	Lisa	1800
NA	Sharon	1350

Pivot table

Area	Sales Person		
	James	Lisa	Sharon
APAC	1750	1500	1850
EMEA	950	1350	2050
NA	1200	1800	1350

Beispiel 3 – Umwandeln von pivotierten Umsatz- und Zieldaten in eine vertikale Tabellenstruktur (fortgeschritten)

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz, der Umsatz- und Zieldaten darstellt, organisiert nach Gebiet und Monat des Jahres. Er wird in eine Tabelle namens „`SalesAndTargets`“ geladen.
- Der Ladezusatz `crosstable`. Er wird verwendet, um die Pivotierung der Dimension `Month Year` aufzuheben und sie in ein dediziertes Feld zu verwandeln und um die Matrix der Umsatz- und Zielbeträge in ein dediziertes Feld namens `Amount` umzuwandeln.
- Eine Konvertierung des Felds `Month Year` zu einem korrekten Datum anhand der Text-zu-Datum-Konvertierungsfunktion `date#`. Dieses datumskonvertierte Feld `Month Year` wird wieder mit der Tabelle `SalesAndTarget` über einen Ladezusatz `Join` verknüpft.

Ladeskript

```
SalesAndTargets:
CROSSTABLE(MonthYearAsText,Amount,2)
LOAD
      *
INLINE [
Area   Type    Jan-22  Feb-22  Mar-22  Apr-22  May-22  Jun-22  Jul-22  Aug-22  Sep-22  Oct-22  Nov-22  Dec-22
APAC   Target  425     425     425     425     425     425     425     425     425     425     425     425
APAC   Actual  435     434     397     404     458     447     413     458     385     421     448     397
```

```
EMEA    Target  362.5   362.5   362.5   362.5   362.5   362.5   362.5   362.5   362.5   362.5   362.5   362.5
EMEA    Actual  363.5   359.5   337.5   361.5   341.5   337.5   379.5   352.5   327.5   337.5   360.5   334.5
NA      Target  375     375     375     375     375     375     375     375     375     375     375     375
NA      Actual  378     415     363     356     403     343     401     365     393     340     360     405
] (delimiter is '\t');

tmp:
LOAD DISTINCT MonthYearAsText,date#(MonthYearAsText,'MMM-YY') AS [Month Year]
RESIDENT SalesAndTargets;

JOIN (SalesAndTargets)
LOAD * RESIDENT tmp;

DROP TABLE tmp;
DROP FIELD MonthYearAsText;
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `Area`
- `Month Year`

Erstellen Sie die folgende Kennzahl mit der Bezeichnung `Actual`:

```
=Sum({<Type={'Actual'}>} Amount)
```

Erstellen Sie zudem die folgende Kennzahl mit der Bezeichnung `Target`:

```
=Sum({<Type={'Target'}>} Amount)
```

Ergebnistabelle (gekürzt)

Fläche	Monat-Jahr	Ist	Ziel
APAC	Jan-22	435	425
APAC	Feb-22	434	425
APAC	Mar-22	397	425
APAC	Apr-22	404	425
APAC	May-22	458	425
APAC	Jun-22	447	425
APAC	Jul-22	413	425
APAC	Aug-22	458	425
APAC	Sep-22	385	425
APAC	Oct-22	421	425

Fläche	Monat-Jahr	Ist	Ziel
APAC	Nov-22	448	425
APAC	Dec-22	397	425
EMEA	Jan-22	363.5	362.5
EMEA	Feb-22	359.5	362.5

Wenn Sie die Anzeige der Daten als pivotierte Eingabetabelle replizieren möchten, können Sie eine entsprechende Pivottabelle in einem Arbeitsblatt erstellen.

Gehen Sie folgendermaßen vor:

1. Kopieren Sie die soeben erstellte Tabelle und fügen Sie sie in ein Arbeitsblatt ein.
2. Ziehen Sie das Diagrammobjekt **Pivottabelle** auf die neu erstellte Tabellenkopie. Wählen Sie **Konvertieren** aus.
3. Klicken Sie auf ✔ **Bearbeitung fertig**.
4. Ziehen Sie das Feld `Month Year` von der Ablage der vertikalen Spalte zur Ablage der horizontalen Spalte.
5. Ziehen Sie das Element `Values` von der Ablage der horizontalen Spalte zur Ablage der vertikalen Spalte.

Die folgende Tabelle zeigt die Daten in ihrer anfänglichen Tabellenform, so wie sie in Qlik Sense angezeigt werden:

Ursprüngliche Ergebnistabelle (gekürzt), wie in Qlik Sense angezeigt

Fläche	Monat-Jahr	Ist	Ziel
Summen	-	13812	13950
APAC	Jan-22	435	425
APAC	Feb-22	434	425
APAC	Mar-22	397	425
APAC	Apr-22	404	425
APAC	May-22	458	425
APAC	Jun-22	447	425
APAC	Jul-22	413	425
APAC	Aug-22	458	425
APAC	Sep-22	385	425
APAC	Oct-22	421	425
APAC	Nov-22	448	425

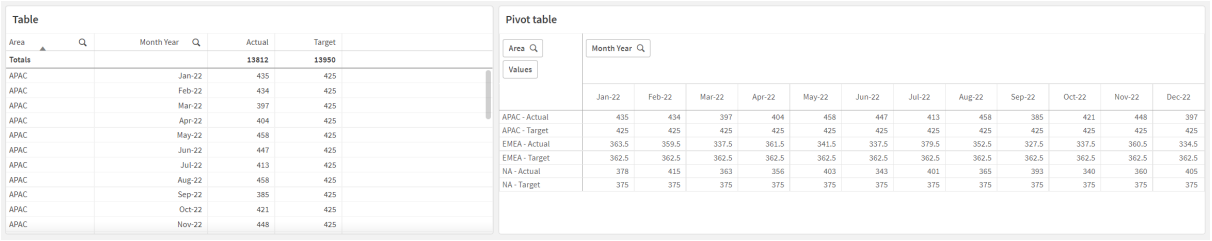
Fläche	Monat-Jahr	Ist	Ziel
APAC	Dec-22	397	425
EMEA	Jan-22	363.5	362.5
EMEA	Feb-22	359.5	362.5

Die entsprechende Pivottabelle gleicht der Folgenden, wobei die Spalte für die einzelnen Monate des Jahres in der längeren Zeile für `Month Year` enthalten ist:

Entsprechende Pivottabelle (gekürzt), wobei das Feld `Month Year` horizontal pivotiert ist

Gebiet (Werte)	Jan-22	Feb-22	Mar-22	Apr-22	May-22	Jun-22	Jul-22	Aug-22	Sep-22	Oct-22	Nov-22	Dec-22
APAC – Ist	435	434	397	404	458	447	413	458	385	421	448	397
APAC – Ziel	425	425	425	425	425	425	425	425	425	425	425	425
EMEA – Ist	363.5	359.5	337.5	361.5	341.5	337.5	379.5	352.5	327.5	337.5	360.5	334.5
EMEA – Ziel	362.5	362.5	362.5	362.5	362.5	362.5	362.5	362.5	362.5	362.5	362.5	362.5
NA – Ist	378	415	363	356	403	343	401	365	393	340	360	405
NA – Ziel	375	375	375	375	375	375	375	375	375	375	375	375

Beispiel der als Tabelle angezeigten Daten und einer entsprechenden Pivottabelle, wobei das Feld `Month Year` horizontal pivotiert ist



First

Der Zusatz `First` vor einem LOAD- oder SELECT-Befehl (SQL) dient dazu, eine festgelegte maximale Anzahl von Datensätzen aus einer Datenquellentabelle zu laden. Ein typischer Anwendungsfall für die Verwendung des Zusatzes `First` ist das Abrufen eines kleinen Teilsatzes von Datensätzen aus einem umfangreichen und/oder langsamen Ladeschritt. Sobald die definierte Zahl von „`n`“ geladen wurde, wird der Ladeschritt vorzeitig abgebrochen, und die restliche Skriptausführung

wird regulär fortgesetzt.

Syntax:

```
First n ( loadstatement | selectstatement )
```

Argumente

Argument	Beschreibung
n	Eine beliebige Formel, die eine ganze Zahl ergibt. Diese Zahl gibt die maximale Anzahl der zu lesenden Datensätze an. n kann auch in Klammern eingeschlossen werden: (n).
loadstatement | selectstatement	Die load statement/select statement, die dem n-Argument folgt, definiert die angegebene Tabelle, die mit der festgelegten maximalen Zahl Datensätze geladen werden muss.

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung SET DateFormat in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Funktionsbeispiele

Beispiel	Ergebnis
FIRST 10 LOAD * from abc.csv;	In diesem Beispiel werden die ersten zehn Zeilen aus einer Excel-Datei abgerufen.
FIRST (1) SQL SELECT * from Orders;	In diesem Beispiel wird die erste ausgewählte Zeile aus dem Datensatz Orders abgerufen.

Beispiel – Laden der ersten fünf Zeilen

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz der Datumswerte für die ersten zwei Wochen von 2020.
- Die Variable `First`, die die Anwendung anweist, nur die ersten fünf Datensätze zu laden.

Ladeskript

```
Sales:
FIRST 5
LOAD
*
Inline [
date,sales
01/01/2020,6000
01/02/2020,3000
01/03/2020,6000
01/04/2020,8000
01/05/2020,5000
01/06/2020,7000
01/07/2020,3000
01/08/2020,5000
01/09/2020,9000
01/10/2020,5000
01/11/2020,7000
01/12/2020,7000
01/13/2020,7000
01/14/2020,7000
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie `Date` als Feld und `sum(sales)` als Kennzahl hinzu.

Ergebnistabelle

Datum	sum(sales)
01/01/2020	6000
01/02/2020	3000
01/03/2020	6000
01/04/2020	8000
01/05/2020	5000

Das Skript lädt nur die ersten fünf Datensätze in die Tabelle `Sales`.

Generic

Das Ladepräfix **Generic** ermöglicht die Konvertierung von Daten eines Element-Attribut-Wert-Modells (Entity-Attribute-Value, EAV) in eine herkömmliche, normalisierte relationale Tabellenstruktur. EAV-Modelle werden alternativ als „generische Datenmodelle“ oder „offenes Schema“ bezeichnet.

Beispiel von Daten nach dem EAV-Modell und einer entsprechenden denormalisierten relationalen Tabelle

Product ID	Attribute	Value
13	Status	Discontinued
13	Colour	Brown
20	Colour	White
13	Size	13-15
20	Size	16-18

Product ID	Status	Colour	Size
13	Discontinued	Brown	13-15
20		White	16-18

Beispiel von Daten nach dem EAV-Modell und einem entsprechenden Satz normalisierter relationaler Tabellen

Product ID	Attribute	Value
13	Status	Discontinued
13	Colour	Brown
20	Colour	White
13	Size	13-15
20	Size	16-18

Product ID	Status
13	Discontinued

Product ID	Colour
13	Brown
20	White

Product ID	Size
13	13-15
20	16-18

Es ist zwar technisch möglich, EAV-Modelldaten in Qlik zu laden und zu analysieren, aber oftmals ist es einfacher, mit einer entsprechenden traditionellen relationalen Datenstruktur zu arbeiten.

Syntax:

```
Generic( loadstatement | selectstatement )
```

Die folgenden Themen können Sie bei der Arbeit mit dieser Funktion unterstützen:

Verwandte Themen

Thema	Beschreibung
Crosstable (page 47)	Der Ladezusatz `Crosstable` wandelt horizontal orientierte Daten in vertikal orientierte Daten um. Aus rein funktionaler Perspektive führt er die entgegengesetzte Umwandlung zum Ladezusatz `Generic` durch, obwohl die Zusätze in der Regel völlig unterschiedlichen Nutzungsfällen dienen.
Generische Datenbanken in *Verwalten von Daten*	EAV-strukturierte Datenmodelle werden hier eingehender beschrieben.

Beispiel 1 – Umwandeln von EAV-strukturierten Daten mit dem Ladezusatz „Generic“

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript enthält einen Datensatz, der in eine Tabelle mit dem Namen `Transactions` geladen wird. Der Datensatz enthält ein Datumsfeld. Die Standarddefinition `MonthNames` wird verwendet.

Ladeskript

```
Products:
Generic
Load * inline [
Product ID, Attribute, Value
13, Status, Discontinued
13, Color, Brown
20, Color, White
13, Size, 13-15
20, Size, 16-18
2, Status, Discontinued
5, Color, Brown
2, Color, White
44, Color, Brown
45, Size, 16-18
45, Color, Brown
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu: `Color`.

Fügen Sie folgende Kennzahl hinzu:

```
=Count([Product ID])
```

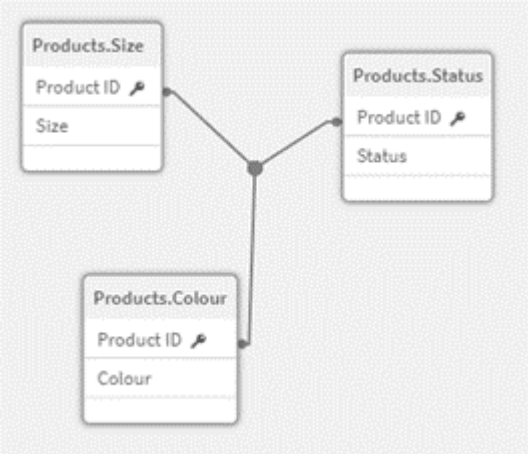
Jetzt können Sie die Anzahl der Produkte nach Farbe prüfen.

Ergebnistabelle

Farbe	**=Count([Product ID])**
Braun	4
Weiß	2

Beachten Sie die Form des Datenmodells, in der jedes Attribut zu einer getrennten Tabelle ausgeweitet wird, die entsprechend dem ursprünglichen Zieltabellen-Tag `Product` benannt wird. Bei jeder Tabelle ist das Attribut ein Suffix. Ein Beispiel hierfür ist `Product.Color`. Die resultierenden „Product Attribute“-Ausgabedatensätze sind nach `Product ID` verknüpft.

Darstellung der Ergebnisse in der Datenmodellansicht



Entstandene Datensatztabelle: Products.Status

Produkt-ID	**Status**
13	Eingestellt
2	Eingestellt

Entstandene Datensatztabelle: Products.Size

Produkt-ID	**Größe**
13	13-15
20	16-18
45	16-18

Entstandene Datensatztabelle: Products.Color

Produkt-ID	**Farbe**
13	Braun
5	Braun

Produkt-ID	Farbe
44	Braun
45	Braun
20	Weiß
2	Weiß

Beispiel 2 – Analysieren von EAV-strukturierten Daten ohne den Ladezusatz „Generic“

Ladeskript und Diagrammformel

Übersicht

Dieses Beispiel zeigt, wie EAV-strukturierte Daten in ihrer ursprünglichen Form analysiert werden.

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript enthält einen Datensatz, der in eine Tabelle mit dem Namen `Products` in einer EAV-Struktur geladen wird.

In diesem Beispiel werden weiter Produkte nach Farbattribut gezählt. Um auf diese Weise strukturierte Daten zu analysieren, müssen Sie Filterung auf Formelebene für Produkte anwenden, die den Attributwert `Color` enthalten.

Zudem können einzelne Attribute nicht als Dimensionen oder Felder ausgewählt werden, was es schwieriger macht zu bestimmen, wie effektive Visualisierungen erstellt werden.

Ladeskript

```
Products:
Load * Inline
[
Product ID, Attribute, Value
13, Status, Discontinued
13, Color, Brown
20, Color, White
13, Size, 13-15
20, Size, 16-18
2, Status, Discontinued
5, Color, Brown
2, Color, White
44, Color, Brown
45, Size, 16-18
45, Color, Brown
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu: `Value`.

Erstellen Sie die folgende Kennzahl:

```
=Count({<Attribute={'Color'}>} [Product ID])
```

Jetzt können Sie die Anzahl der Produkte nach Farbe prüfen.

Entstandene Datensatztabelle: Products.Status

Wert	**=Count({<Attribute={'Color'}>} [Product ID])**
Braun	4
Weiß	2

Beispiel 3 – Denormalisieren der entstandenen Ausgabetabellen aus einem „Generic“-Ladevorgang (fortgeschritten)

Ladeskript und Diagrammformel

Übersicht

In diesem Beispiel wird gezeigt, wie die normalisierte Datenstruktur, die von dem Ladezusatz `Generic` produziert wird, zurück in eine konsolidierte `Product`-Dimensionstabelle denormalisiert werden kann. Dies ist eine erweiterte Modellerstellungstechnik, die als Teil der Feinabstimmung der Leistung des Datenmodells verwendet werden kann.

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Ladeskript

```
Products:

Generic
Load * inline [
Product ID, Attribute, Value
13, Status, Discontinued
13, Color, Brown
20, Color, White
13, Size, 13-15
20, Size, 16-18
2, Status, Discontinued
5, Color, Brown
2, Color, White
44, Color, Brown
45, Size, 16-18
45, Color, Brown
];
```

```
RENAME TABLE Products.Color TO Products;

OUTER JOIN (Products)
LOAD * RESIDENT Products.Size;

OUTER JOIN (Products)
LOAD * RESIDENT Products.Status;
DROP TABLES Products.Size,Products.Status;
```

Ergebnisse

Öffnen Sie die Datenmodellansicht und beachten Sie die Form des entstandenen Datenmodells. Nur eine denormalisierte Tabelle ist vorhanden. Es handelt sich um eine Kombination der drei Zwischen-Ausgabetabellen: `Products.Size`, `Products.Status` und `Products.Color`.

Entstandenes internes Datenmodell

Produkte
Produkt-ID
Status
Farbe
Größe

Entstandene Datensatztabelle: Produkte

Produkt-ID	**Status**	**Farbe**	**Größe**
13	Eingestellt	Braun	13-15
20	-	Weiß	16-18
2	Eingestellt	Weiß	-
5	-	Braun	-
44	-	Braun	-
45	-	Braun	16-18

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu: `Color`.

Fügen Sie folgende Kennzahl hinzu:

```
=Count([Product ID])
```

Ergebnistabelle

Farbe	**=Count([Product ID])**
Braun	4
Weiß	2

Hierarchy

Mithilfe des **hierarchy**-Zusatzes wird eine über-/untergeordnete Hierarchie-Tabelle in eine Tabelle umgewandelt, die in einem Qlik Sense-Datenmodell verwendet werden kann. Er kann vor dem Befehl **LOAD** oder **SELECT** eingefügt werden und verwendet anschließend das Ergebnis des Ladebefehls als Eingabe für eine Tabellenumformung.

Der Zusatz erstellt eine erweiterte Knotentabelle, die normalerweise über die gleiche Anzahl von Datensätzen wie die Eingabetabelle verfügt. Darüber hinaus wird jedoch jeder Produktlevel in der Hierarchie in einem gesonderten Feld gespeichert. Das Pfadfeld kann in einer Baumstruktur verwendet werden.

Syntax:

```
Hierarchy (NodeID, ParentID, NodeName, [ParentName, [PathSource, [PathName,
[PathDelimiter, Depth]]]])(loadstatement | selectstatement)
```

Die Eingabetabelle muss eine Tabelle mit benachbarten Knoten sein. Tabellen mit benachbarten Knoten sind Tabellen, in denen jeder Datensatz einem Knoten entspricht und über ein Feld verfügt, das auf einen übergeordneten Knoten verweist. Jeder Knoten wird in dieser Tabelle in nur einem Datensatz gespeichert, jedoch kann der Knoten über mehrere untergeordnete Elemente verfügen. Natürlich kann die Tabelle auch weitere Felder enthalten, etwa mit Attributen.

Der Zusatz erstellt eine erweiterte Knotentabelle, die normalerweise über die gleiche Anzahl von Datensätzen wie die Eingabetabelle verfügt. Darüber hinaus wird jedoch jeder Produktlevel in der Hierarchie in einem gesonderten Feld gespeichert. Das Pfadfeld kann in einer Baumstruktur verwendet werden.

Normalerweise enthält die interne Tabelle für jeden Knoten genau einen Datensatz. In einem derartigen Fall enthält die Ausgabetabelle die gleiche Anzahl an Datensätzen. Unter Umständen liegen auch Knoten mit mehreren übergeordneten Tabellen vor, d. h. ein Knoten wird mithilfe mehrerer Datensätze in der Eingabetabelle dargestellt. In diesem Fall enthält die Ausgabetabelle u. U. mehr Datensätze als die Eingabetabelle.

Alle Knoten mit einer ElternID, die nicht in der Spalte KnotenID enthalten ist (einschließlich Knoten mit einer fehlenden ElternID) gelten als Wurzelknoten. Um Zirkelbezüge zu vermeiden, werden nur Knoten geladen, die eine direkte oder indirekte Verbindung zu einem Wurzelknoten haben.

Es können zusätzliche Felder für Name des Elternknotens, Pfad und Ebene des Knotens generiert werden.

Argumente:

Argumente

Argument	Beschreibung
NodeID	Der Name des Felds, das die Knoten-ID enthält. Das Feld muss in der Eingabetabelle vorhanden sein.
ParentID	Der Name des Felds, das die ID der Eltern, d. h. des direkt übergeordneten Knotens, enthält. Das Feld muss in der Eingabetabelle vorhanden sein.
NodeName	Der Name des Felds, das die Bezeichnung des Knotens enthält. Das Feld muss in der Eingabetabelle vorhanden sein.
ParentName	Ein String zur Benennung des neuen **ParentName**-Felds. Fehlt dieser Parameter, wird kein solches Feld generiert.
ParentSource	Der Name des Felds, das die Namen der Knoten enthält, die für den Pfad zum Knoten verwendet werden sollen. Dieser Parameter ist optional. Wird er weggelassen, wird **NodeName** verwendet.
PathName	Ein String zur Benennung des neuen **Path**-Feldes, das den Pfad vom Wurzelknoten zum betreffenden Knoten enthält. Dieser Parameter ist optional. Fehlt dieser Parameter, wird kein solches Feld generiert.
PathDelimiter	Ein String, der als Trennzeichen im neuen **Path**-Feld dient. Dieser Parameter ist optional. Wird er weggelassen, wird '/' verwendet.
Depth	Ein String zur Benennung des neuen **Depth**-Felds, das angibt, in welcher Ebene, vom Wurzelknoten aus betrachtet, sich der betreffende Knoten befindet. Dieser Parameter ist optional. Fehlt dieser Parameter, wird kein solches Feld generiert.

Beispiel:

```
Hierarchy(NodeID, ParentID, NodeName, ParentName, NodeName, PathName, '\', Depth) LOAD *
inline [

NodeID, ParentID, NodeName

1, 4, London

2, 3, Munich

3, 5, Germany

4, 5, UK

5, , Europe

];
```

NodeID	ParentID	NodeName	NodeName1	NodeName2	NodeName3	ParentName	PathName	Depth
1	4	London	Europe	UK	London	UK	Europe\UK\London	3
2	3	Munich	Europe	Germany	Munich	Germany	Europe\Germany\Munich	3
3	5	Germany	Europe	Germany	-	Europe	Europe\Germany	2
4	5	UK	Europe	UK	-	Europe	Europe\UK	2
5		Europe	Europe	-	-	-	Europe	1

HierarchyBelongsTo

Mithilfe dieses Zusatzes wird eine über-/untergeordnete Hierarchie-Tabelle in eine Tabelle umgewandelt, die in einem Qlik Sense-Datenmodell verwendet werden kann. Er kann vor dem Befehl **LOAD** oder **SELECT** eingefügt werden und verwendet anschließend das Ergebnis des Ladebefehls als Eingabe für eine Tabellenumformung.

Die resultierende Vorfahrentabelle enthält für jeden Knoten sämtliche Vorfahren, d. h. alle übergeordneten Knoten. Das Feld mit den Vorfahren eignet sich besonders gut für Selektionen innerhalb der Hierarchie. Die Ausgabetabelle enthält normalerweise mehrere Datensätze pro Knoten.

Syntax:

```
HierarchyBelongsTo (NodeID, ParentID, NodeName, AncestorID, AncestorName,
[DepthDiff])(loadstatement | selectstatement)
```

Die Eingabetabelle muss eine Tabelle mit benachbarten Knoten sein. Tabellen mit benachbarten Knoten sind Tabellen, in denen jeder Datensatz einem Knoten entspricht und über ein Feld verfügt, das auf einen übergeordneten Knoten verweist. Jeder Knoten wird in dieser Tabelle in nur einem Datensatz gespeichert, jedoch kann der Knoten über mehrere untergeordnete Elemente verfügen. Natürlich kann die Tabelle auch weitere Felder enthalten, etwa mit Attributen.

Die resultierende Vorfahrentabelle enthält für jeden Knoten sämtliche Vorfahren, d. h. alle übergeordneten Knoten. Das Feld mit den Vorfahren eignet sich besonders gut für Selektionen innerhalb der Hierarchie. Die Ausgabetabelle enthält normalerweise mehrere Datensätze pro Knoten.

Es kann ein zusätzliches Feld für die Zahl der Ebenen zwischen den Knoten generiert werden.

Argumente:

Argumente

Argument	Beschreibung
NodeID	Der Name des Felds, das die Knoten-ID enthält. Das Feld muss in der Eingabetabelle vorhanden sein.
ParentID	Der Name des Felds, das die ID der Eltern, d. h. des direkt übergeordneten Knotens, enthält. Das Feld muss in der Eingabetabelle vorhanden sein.
NodeName	Der Name des Felds, das die Bezeichnung des Knotens enthält. Das Feld muss in der Eingabetabelle vorhanden sein.
AncestorID	Ein String zur Benennung des neuen Vorfahr ID-Felds, das die ID der Vorfahren enthält.
AncestorName	Ein String zur Benennung des neuen Vorfahr-Felds, das den Namen der Vorfahren enthält.
DepthDiff	Ein String zur Benennung des neuen **DepthDiff**-Felds, das die Zahl der Ebenen zwischen dem Knoten und dem Vorfahr enthält. Dieser Parameter ist optional. Fehlt dieser Parameter, wird kein solches Feld generiert.

Beispiel:

```
HierarchyBelongsTo (NodeID, AncestorID, NodeName, AncestorID, AncestorName, DepthDiff) LOAD *
inline [

NodeID, AncestorID, NodeName

1, 4, London

2, 3, Munich

3, 5, Germany

4, 5, UK

5, , Europe

];
```

Results

NodeID	AncestorID	NodeName	AncestorName	DepthDiff
1	1	London	London	0
1	4	London	UK	1

NodeID	AncestorID	NodeName	AncestorName	DepthDiff
1	5	London	Europe	2
2	2	Munich	Munich	0
2	3	Munich	Germany	1
2	5	Munich	Europe	2
3	3	Germany	Germany	0
3	5	Germany	Europe	1
4	4	UK	UK	0
4	5	UK	Europe	1
5	5	Europe	Europe	0

Inner

Dem Präfix **join** und dem Präfix **keep** kann das Präfix **inner** vorangestellt sein. Vor einem **join**-Befehl bestimmt dieser Zusatz, dass ein Inner Join hergestellt werden soll. Die entstandene Datei enthält somit lediglich Kombinationen von Feldwerten der Datentabellen, wobei die Referenzen zu externen Werten in beiden Tabellen dargestellt werden. Bei Verwendung vor **keep** bestimmt dieser Zusatz, dass beide Rohdatentabellen auf ihre gemeinsame Schnittmenge reduziert werden sollen, bevor sie in Qlik Sense gespeichert werden.

Syntax:

```
Inner ( Join | Keep) [ (tablename) ](loadstatement |selectstatement )
```

Argumente:

Argumente

Argument	Beschreibung
tablename	Die benannte Tabelle, die mit der geladenen Tabelle verglichen wird.
loadstatementoder selectstatement	Der **LOAD**- oder **SELECT**-Befehl für die geladene Tabelle.

Beispiel

Ladeskript

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Fügen Sie einem Arbeitsblatt in Ihrer App dann die Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um das Ergebnis anzuzeigen.

```
Table1: Load * inline [ Column1, Column2 A, B 1, aa 2, cc 3, ee ];  Table2: Inner Join Load *
inline [ Column1, Column3 A, C 1, xx 4, yy ];
```

Ergebnis

Ergebnistabelle

Column1	Column2	Column3
A	B	C
1	aa	xx

Erläuterung

Dieses Beispiel zeigt die Ausgabe für den Inner Join, wenn Werte, die sowohl in der ersten (linken) als auch in der zweiten (rechten) Tabelle vorhanden sind, verknüpft werden.

IntervalMatch

Mithilfe des Zusatzes **IntervalMatch** wird eine Tabelle angelegt, in der die diskreten numerischen Werte mit einem oder mehreren numerischen Intervallen abgeglichen werden und optional die Werte von einem oder mehreren zusätzlichen Schlüsseln.

Syntax:

```
IntervalMatch (matchfield)(loadstatement | selectstatement )
```

```
IntervalMatch (matchfield,keyfield1 [ , keyfield2, ... keyfield5 ] )
(loadstatement | selectstatement )
```

Der Zusatz **IntervalMatch** muss vor dem Befehl **LOAD** oder **SELECT** angelegt werden, mit dem die Intervalle geladen werden. Das Feld, das die diskreten Datenpunkte (im Beispiel unten die Uhrzeit) und zusätzlichen Schlüssel enthält, muss vor dem Befehl mit dem **IntervalMatch**-Zusatz bereits in Qlik Sense geladen sein. Der Zusatz liest dieses Feld nicht selbstständig in der Datenbanktabelle. Der Zusatz formt die geladene Intervalltabelle und Schlüssel in eine Tabelle um, die eine zusätzliche Spalte enthält: die diskreten numerischen Datenpunkte. Er erweitert auch die Anzahl der Datensätze, sodass die neue Tabelle über einen Datensatz für jede mögliche Kombination diskreter Datenpunkte, des Intervalls und des Werts der Schlüsselfelder verfügt.

Wenn sich die Intervalle überschneiden, werden die Werte jedem passenden Intervall zugeordnet.

Wenn der IntervalMatch -Zusatz mit Schlüsselfeldern erweitert wird, wird damit eine Tabelle mit abgeglichenen numerischen Werten für ein oder mehrere numerische Intervalle erzeugt, wobei gleichzeitig die Werte eines oder mehrerer zusätzlicher Schlüssel abgeglichen werden.

Um zu verhindern, dass nicht definierte Intervallbeschränkungen ignoriert werden, sollte es möglich sein, NULL-Werte anderen Feldern zuzuordnen, welche die untere oder obere Grenze des Intervalls bilden. Dieser Vorgang kann über den Befehl **NullAsValue** oder einen Test erfolgen, in dem NULL-Werte deutlich vor oder nach den diskreten numerischen Datenpunkten durch einen numerischen Wert ersetzt werden.

Argumente:

Argumente

Argument	Beschreibung
matchfield	Das Feld, dessen numerische Werte in Intervallen gruppiert werden.
keyfield	Felder, welche die zusätzlichen Attribute enthalten, die während der Umformung abgeglichen werden.
loadstatement orselectstatement	Muss in einer Tabelle resultieren, in der das erste Feld den unteren Grenzwert jedes Intervalls, das zweite Feld den oberen Grenzwert jedes Intervalls und – bei Verwendung von Schlüsselwortabgleich – das dritte und alle folgenden Felder das/die Schlüsselfeld(er) enthalten, die im **IntervalMatch**-Befehl enthalten sind. Die Intervalle sind abgeschlossen, d. h. die Grenzwerte sind in den Intervallen enthalten. Nicht numerische Beschränkungen bewirken, dass das Intervall nicht berücksichtigt wird (undefiniert).

Example 1:

In den beiden unten stehenden Tabellen zeigt die erste Tabelle eine Reihe diskreter Ereignisse und die zweite den Bearbeitungsstart und das Bearbeitungsende verschiedener Aufträge an. Mithilfe des Zusatzes **IntervalMatch** werden die Tabellen logisch verknüpft, um z. B. herauszufinden, welche Aufträge von welchen Ereignissen betroffen waren und welche Aufträge von welcher Schicht verarbeitet wurden.

```
EventLog:
LOAD * Inline [
Time, Event, Comment
00:00, 0, Start of shift 1
01:18, 1, Line stop
02:23, 2, Line restart 50%
04:15, 3, Line speed 100%
08:00, 4, Start of shift 2
11:43, 5, End of production
];

OrderLog:
LOAD * INLINE [
Start, End, Order
01:00, 03:35, A
02:30, 07:58, B
03:04, 10:27, C
07:23, 11:43, D
];

//Link the field Time to the time intervals defined by the fields Start and End.
Inner Join IntervalMatch ( Time )
LOAD Start, End
Resident OrderLog;
```

Die Tabelle**OrderLog** enthält nun eine zusätzliche Spalte: *Time*. Der Anzahl der Datensätze wird ebenfalls erweitert.

Table with additional column

Time	Start	End	Order
00:00	-	-	-
01:18	01:00	03:35	A
02:23	01:00	03:35	A
04:15	02:30	07:58	B
04:15	03:04	10:27	C
08:00	03:04	10:27	C
08:00	07:23	11:43	D
11:43	07:23	11:43	D

Example 2: (verwenden von keyfield)

Gleiches Beispiel wie oben, zusätzlich mit *ProductionLine* als Schlüsselfeld.

```
EventLog:

LOAD * Inline [

Time, Event, Comment, ProductionLine

00:00, 0, Start of shift 1, P1

01:00, 0, Start of shift 1, P2

01:18, 1, Line stop, P1

02:23, 2, Line restart 50%, P1

04:15, 3, Line speed 100%, P1

08:00, 4, Start of shift 2, P1

09:00, 4, Start of shift 2, P2

11:43, 5, End of production, P1

11:43, 5, End of production, P2

];


OrderLog:
```

```
LOAD * INLINE [

Start, End, Order, ProductionLine

01:00, 03:35, A, P1

02:30, 07:58, B, P1

03:04, 10:27, C, P1

07:23, 11:43, D, P2

];



//Link the field Time to the time intervals defined by the fields Start and End and match the
values

// to the key ProductionLine.

Inner Join

IntervalMatch ( Time, ProductionLine )

LOAD Start, End, ProductionLine

Resident OrderLog;
```

Nun kann die folgende Tabellenbox erzeugt werden:

Tablebox example

ProductionLine	Time	Event	Comment	Order	Start	End
P1	00:00	0	Start of shift 1	-	-	-
P2	01:00	0	Start of shift 1	-	-	-
P1	01:18	1	Line stop	A	01:00	03:35
P1	02:23	2	Line restart 50%	A	01:00	03:35
P1	04:15	3	Line speed 100%	B	02:30	07:58
P1	04:15	3	Line speed 100%	C	03:04	10:27
P1	08:00	4	Start of shift 2	C	03:04	10:27
P2	09:00	4	Start of shift 2	D	07:23	11:43
P1	11:43	5	End of production	-	-	-
P2	11:43	5	End of production	D	07:23	11:43

Join

Der **join**-Zusatz kombiniert die geladene Tabelle mit einer bestehenden benannten Tabelle bzw. mit der zuletzt erstellten Datentabelle.

Die Auswirkung der Verknüpfung von Daten besteht darin, die Zieltabelle durch einen weiteren Satz Felder oder Attribute zu erweitern, nämlich diejenigen, die in der Zieltabelle noch nicht vorhanden sind. Jeder gemeinsame Feldname zwischen dem Quelldatensatz und der Zieltabelle wird dazu verwendet, zu ermitteln, wie die neuen eingehenden Datensätze verknüpft werden. Dies wird in der Regel als „Natural Join“ bezeichnet. Ein QlikJoin-Vorgang kann dazu führen, dass die entstehende Tabelle mehr oder weniger Datensätze als anfänglich hat, abhängig von der Eindeutigkeit der Join-Verknüpfung und dem Typ des verwendeten Joins.

Es gibt vier Join-Typen:

Left Join

Left Joins sind der üblichste Join-Typ. Wenn Sie beispielsweise einen Transaktionsdatensatz mit einem Referenzdatensatz kombinieren möchten, verwenden Sie in der Regel einen `Left Join`. In diesem Fall laden Sie zuerst die Transaktionstabelle, dann den Referenzdatensatz, während Sie ihn über einen `Left Join`-Zusatz mit der bereits geladenen Transaktionstabelle verknüpfen. Ein `Left Join` behält alle Transaktionen unverändert bei und fügt die ergänzenden Referenzdatenfelder hinzu, wenn eine Übereinstimmung gefunden wird.

Inner Join

Wenn Sie zwei Datensätze haben, bei denen Sie nur an Ergebnissen interessiert sind, wenn eine übereinstimmende Verknüpfung vorliegt, können Sie einen `Inner Join` verwenden. Dadurch werden alle Datensätze sowohl aus den geladenen Quelldaten als auch aus der Zieltabelle entfernt, wenn keine Übereinstimmung gefunden wird. Als Ergebnis enthält Ihre Zieltabelle dann möglicherweise weniger Datensätze als vor dem Join-Vorgang.

Outer Join

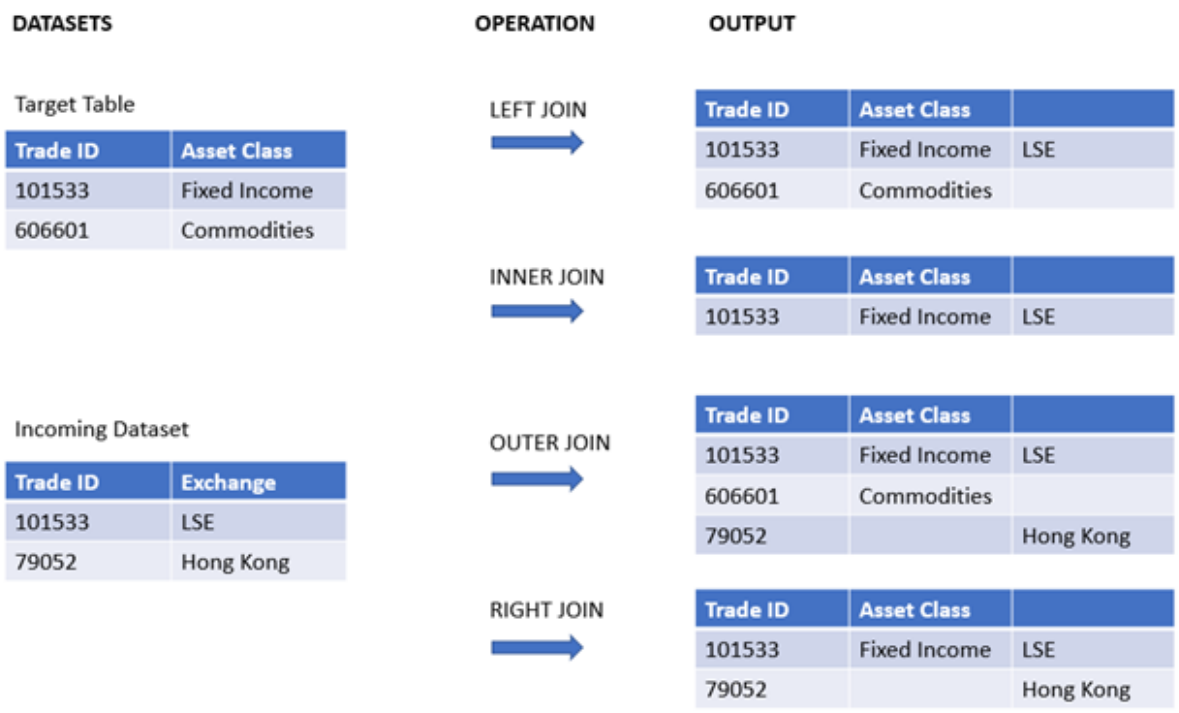
Wenn Sie sowohl die Zieldatensätze als auch alle eingehenden Datensätze beibehalten möchten, verwenden Sie einen `Outer Join`. Wo keine Übereinstimmung gefunden wird, wird jeder Satz Datensätze dennoch beibehalten, während die Felder auf der gegenüber liegenden Seite des Join unausgefüllt (null) bleiben.

Wenn das Schlüsselwort für den Typ ausgelassen wird, ist Outer Join der Standard-Join-Typ.

Right Join

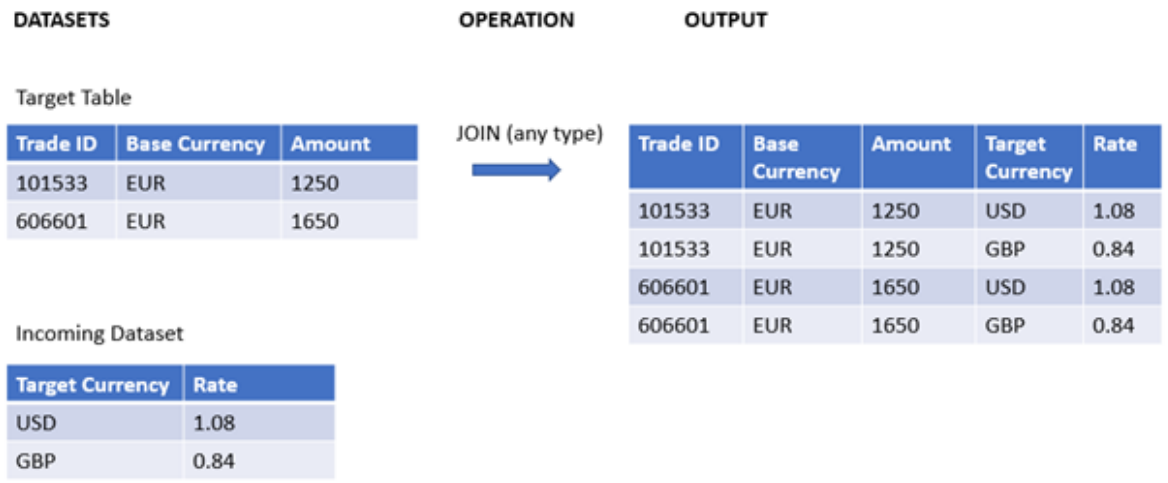
Bei diesem Join-Typ werden alle zu ladenden Datensätze beibehalten, während die Datensätze in der Zieltabelle des Join auf diejenigen reduziert werden, bei denen eine Übereinstimmung bei den eingehenden Datensätzen vorliegt. Dies ist ein Nischen-Join-Typ, der manchmal als Möglichkeit zum Verkleinern einer bereits geladenen Datensatztabelle auf einen erforderlichen Teilsatz verwendet wird.

Beispiel-Ergebnissätze nach verschiedenen Typen von Join-Vorgängen



Wenn die Quelle und das Ziel eines Join-Vorgangs keine gemeinsamen Feldnamen enthalten, ergibt der Join ein kartesianisches Produkt aller Zeilen – dies wird als „Cross Join“ bezeichnet.

Ergebnisbeispiel für einen „Cross Join“-Vorgang.



Syntax:

```
[inner | outer | left | right ]Join [ (tablename ) ]( loadstatement | selectstatement )
```

Argumente

Argument	Beschreibung
tablename	Die benannte Tabelle, die mit der geladenen Tabelle verglichen wird.
loadstatementoder selectstatement	Der **LOAD**- oder **SELECT**-Befehl für die geladene Tabelle.

Die folgenden Themen können Sie bei der Arbeit mit dieser Funktion unterstützen:

Verwandte Themen

Thema	Beschreibung
Kombinieren von Tabellen mit Join und Keep in *Verwalten von Daten*	Dieses Thema enthält weitere Erläuterungen der Konzepte von „Join“ (Verknüpfen) und „Keep“ (Beibehalten) von Datensätzen.
Keep (page 84)	Der Ladezusatz `Keep` gleicht dem Zusatz `Join`, kombiniert aber die Quell- und Zieldatensätze nicht. Stattdessen kürzt er jeden Datensatz entsprechend dem gewählten Vorgangstyp (Inner, Outer, Left oder Right).

Beispiel 1 – Left Join: Erweitern einer Zieltabelle mit einem Referenzdatensatz

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz mit Änderungsdatensätzen, der in eine Tabelle namens `Changes` geladen wird. Sie enthält ein „Status ID“-Schlüsselfeld.
- Ein zweiter Datensatz, der Änderungsstatus darstellt, wird geladen und mit den ursprünglichen Änderungsdatensätzen kombiniert, indem sie mit dem Ladezusatz „Left `Join`“ verknüpft werden.

Dieser Left Join sorgt dafür, dass die Änderungsdatensätze intakt bleiben, während Statusattribute hinzugefügt werden, wenn eine Übereinstimmung mit den eingehenden Statuseinträgen basierend auf einer gemeinsamen Status-ID gefunden wird.

Ladeskript

```
Changes:
Load * inline [
Change ID      Status ID       Scheduled Start Date    Scheduled End Date      Business Impact
10030  4       19/01/2022      23/02/2022      None
10015  3       04/01/2022      15/02/2022      Low
10103  1       02/04/2022      29/05/2022      Medium
10185  2       23/06/2022      08/09/2022      None
```

```
10323  1       08/11/2022      26/11/2022      High
10326  2       11/11/2022      05/12/2022      None
10138  2       07/05/2022      03/08/2022      None
10031  3       20/01/2022      25/03/2022      Low
10040  1       29/01/2022      22/04/2022      None
10134  1       03/05/2022      08/07/2022      Low
10334  2       19/11/2022      06/02/2023      Low
10220  2       28/07/2022      06/09/2022      None
10264  1       10/09/2022      17/10/2022      Medium
10116  1       15/04/2022      24/04/2022      None
10187  2       25/06/2022      24/08/2022      Low
] (delimiter is '\t');

Status:
Left Join (Changes)
Load * inline [
Status ID      Status  Sub Status
1      Open    Not Started
2      Open    Started
3      Closed  Completed
4      Closed  Cancelled
5      Closed  Obsolete
] (delimiter is '\t');
```

Ergebnisse

Öffnen Sie die Datenmodellansicht und beachten Sie die Form des Datenmodells. Nur eine denormalisierte Tabelle ist vorhanden. Es handelt sich um eine Kombination aller ursprünglichen Änderungsdatensätze, wobei die übereinstimmenden Statusattribute jedem Änderungsdatensatz hinzugefügt werden.

Entstandenes internes Datenmodell

Änderungen
Änderungs-ID
Status-ID
Geplantes Startdatum
Geplantes Enddatum
Geschäftliche Auswirkung
Status
Unterstatus

Wenn Sie das Vorschaufenster in der Datenmodellansicht erweitern, sehen Sie einen Teil dieses vollständigen Ergebnissatzes in Form einer Tabelle:

Vorschau der Änderungstabelle in der Datenmodellansicht

Änderungs-ID	Status-ID	Geplantes Startdatum	Geplantes Enddatum	Geschäftliche Auswirkung	Status	Unterstatus
10030	4	19/01/2022	23/02/2022	Keine	Geschlossen	Abgebrochen
10031	3	20/01/2022	25/03/2022	Niedrig	Geschlossen	Abgeschlossen
10015	3	04/01/2022	15/02/2022	Niedrig	Geschlossen	Abgeschlossen
10103	1	02/04/2022	29/05/2022	Mittel	Öffnen	Nicht gestartet
10116	1	15/04/2022	24/04/2022	Keine	Öffnen	Nicht gestartet
10134	1	03/05/2022	08/07/2022	Niedrig	Öffnen	Nicht gestartet
10264	1	10/09/2022	17/10/2022	Mittel	Öffnen	Nicht gestartet
10040	1	29/01/2022	22/04/2022	Keine	Öffnen	Nicht gestartet
10323	1	08/11/2022	26/11/2022	Hoch	Öffnen	Nicht gestartet
10187	2	25/06/2022	24/08/2022	Niedrig	Öffnen	Gestartet
10185	2	23/06/2022	08/09/2022	Keine	Öffnen	Gestartet
10220	2	28/07/2022	06/09/2022	Keine	Öffnen	Gestartet
10326	2	11/11/2022	05/12/2022	Keine	Öffnen	Gestartet
10138	2	07/05/2022	03/08/2022	Keine	Öffnen	Gestartet
10334	2	19/11/2022	06/02/2023	Niedrig	Öffnen	Gestartet

Da die fünfte Zeile in der Statustabelle (Status-ID: „5“, Status: „Geschlossen“, Unterstatus: „Veraltet“) keinem der Datensätze in der Änderungstabelle entspricht, werden die Informationen dieser Zeile nicht im obigen Ergebnissatz angezeigt.

Kehren Sie zum Dateneditor zurück. Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu: `Status`.

Fügen Sie folgende Kennzahl hinzu:

```
=Count([Change ID])
```

Jetzt können Sie die Anzahl der Änderungen nach Status prüfen.

Ergebnistabelle

Status	=Count([Change ID])
Öffnen	12
Geschlossen	3

Beispiel 2 – Inner Join: Nur Kombinieren übereinstimmender Datensätze

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz mit Änderungsdatensätzen, der in eine Tabelle namens `Changes` geladen wird.
- Ein zweiter Datensatz, der Änderungsdatensätze aus dem Quellsystem `JIRA` darstellt, wird geladen und mit den ursprünglichen Datensätzen kombiniert, indem sie mit dem Ladezusatz `Inner Join` verknüpft werden.

Dieser `Inner Join` sorgt dafür, dass nur die fünf in beiden Datensätzen gefundenen Änderungsdatensätze beibehalten werden.

Ladeskript

```
Changes:
Load * inline [
Change ID      Status ID       Scheduled Start Date    Scheduled End Date      Business Impact
10030  4       19/01/2022      23/02/2022      None
10015  3       04/01/2022      15/02/2022      Low
10103  1       02/04/2022      29/05/2022      Medium
10185  2       23/06/2022      08/09/2022      None
10323  1       08/11/2022      26/11/2022      High
10326  2       11/11/2022      05/12/2022      None
10138  2       07/05/2022      03/08/2022      None
10031  3       20/01/2022      25/03/2022      Low
10040  1       29/01/2022      22/04/2022      None
10134  1       03/05/2022      08/07/2022      Low
10334  2       19/11/2022      06/02/2023      Low
10220  2       28/07/2022      06/09/2022      None
10264  1       10/09/2022      17/10/2022      Medium
10116  1       15/04/2022      24/04/2022      None
10187  2       25/06/2022      24/08/2022      Low
] (delimiter is '\t');

JIRA_changes:
Inner Join (Changes)
Load
  [Ticket ID] AS [Change ID],
  [Source System]
inline
[
Ticket ID      Source System
10000  JIRA
10030  JIRA
10323  JIRA
10134  JIRA
10334  JIRA
```

```
10220  JIRA
20000  TFS
] (delimiter is '\t');
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `Source System`
- `Change ID`
- `Business Impact`

Jetzt können Sie die fünf Ergebnisdatensätze prüfen. Die Ergebnistabelle eines `Inner Join` enthält nur Einträge mit übereinstimmenden Informationen in beiden Datensätzen.

Ergebnistabelle

Quellsystem	Änderungs-ID	Geschäftliche Auswirkung
JIRA	10030	Keine
JIRA	10134	Niedrig
JIRA	10220	Keine
JIRA	10323	Hoch
JIRA	10334	Niedrig

Beispiel 3 – Outer Join: Kombinieren von überlappenden Datensätzen

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz mit Änderungsdatensätzen, der in eine Tabelle namens `Changes` geladen wird.
- Ein zweiter Datensatz, der Änderungsdatensätze aus dem Quellsystem `JIRA` darstellt, wird geladen und mit den ursprünglichen Datensätzen kombiniert, indem sie mit dem Ladezusatz `Outer Join` verknüpft werden.

Dies sorgt dafür, dass alle überlappenden Änderungsdatensätze aus beiden Datensätzen beibehalten werden.

Ladeskript

```
// 8 Change records
```

```
Changes:
Load * inline [
Change ID      Status ID       Scheduled Start Date    Scheduled End Date      Business Impact
10030  4       19/01/2022      23/02/2022      None
10015  3       04/01/2022      15/02/2022      Low
10138  2       07/05/2022      03/08/2022      None
10031  3       20/01/2022      25/03/2022      Low
10040  1       29/01/2022      22/04/2022      None
10134  1       03/05/2022      08/07/2022      Low
10334  2       19/11/2022      06/02/2023      Low
10220  2       28/07/2022      06/09/2022      None
] (delimiter is '\t');

// 6 Change records

JIRA_changes:
Outer Join (Changes)
Load
  [Ticket ID] AS [Change ID],
  [Source System]
inline
[
Ticket ID      Source System
10030  JIRA
10323  JIRA
10134  JIRA
10334  JIRA
10220  JIRA
10597  JIRA
] (delimiter is '\t');
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `Source System`
- `Change ID`
- `Business Impact`

Jetzt können Sie die zehn Ergebnisdatensätze prüfen.

Ergebnistabelle

Quellsystem	Änderungs-ID	Geschäftliche Auswirkung
JIRA	10030	Keine
JIRA	10134	Niedrig
JIRA	10220	Keine
JIRA	10323	-

Quellsystem	Änderungs-ID	Geschäftliche Auswirkung
JIRA	10334	Niedrig
JIRA	10597	-
-	10015	Niedrig
-	10031	Niedrig
-	10040	Keine
-	10138	Keine

Beispiel 4 – Right Join: Kürzen einer Zieltabelle durch einen sekundären Master-Datensatz

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz mit Änderungsdatensätzen, der in eine Tabelle namens `Changes` geladen wird.
- Ein zweiter Datensatz stellt die Änderungsdatensätze dar, die aus dem Quellsystem `Teamwork` stammen. Diese werden geladen und mit den Originaldatensätzen kombiniert, indem sie mit einem `Right Join`-Ladepräfix verknüpft werden.

Dies sorgt dafür, dass nur `Teamwork`-Änderungsdatensätze beibehalten werden und gleichzeitig keine `Teamwork`-Datensätze verloren gehen, wenn die Zieltabelle keine übereinstimmende `Change ID` enthält.

Ladeskript

`Changes:`

```
Load * inline [
Change ID      Status ID       Scheduled Start Date    Scheduled End Date      Business Impact
10030  4       19/01/2022      23/02/2022      None
10015  3       04/01/2022      15/02/2022      Low
10103  1       02/04/2022      29/05/2022      Medium
10185  2       23/06/2022      08/09/2022      None
10323  1       08/11/2022      26/11/2022      High
10326  2       11/11/2022      05/12/2022      None
10138  2       07/05/2022      03/08/2022      None
10031  3       20/01/2022      25/03/2022      Low
10040  1       29/01/2022      22/04/2022      None
10134  1       03/05/2022      08/07/2022      Low
10334  2       19/11/2022      06/02/2023      Low
10220  2       28/07/2022      06/09/2022      None
10264  1       10/09/2022      17/10/2022      Medium
10116  1       15/04/2022      24/04/2022      None
```

```
10187  2       25/06/2022      24/08/2022      Low
] (delimiter is '\t');

Teamwork_changes:
Right Join (Changes)
Load
  [Ticket ID] AS [Change ID],
  [Source System]
inline
[
Ticket ID      Source System
10040  Teamwork
10015  Teamwork
10103  Teamwork
10031  Teamwork
50231  Teamwork
] (delimiter is '\t');
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `Source System`
- `Change ID`
- `Business Impact`

Jetzt können Sie die fünf Ergebnisdatensätze prüfen.

Ergebnistabelle

Quellsystem	**Änderungs-ID**	**Geschäftliche Auswirkung**
Teamwork	10015	Niedrig
Teamwork	10031	Niedrig
Teamwork	10040	Keine
Teamwork	10103	Mittel
Teamwork	50231	-

Keep

Der Zusatz **keep** weist Ähnlichkeiten mit dem Zusatz **join** auf. Wie der **join**-Zusatz kombiniert er die geladene Tabelle mit einer bestehenden benannten Tabelle oder der zuletzt erstellten Datentabelle, doch statt die geladene Tabelle mit einer bestehenden Tabelle zusammenzuschließen, bewirkt der Zusatz, dass die Tabelle oder beide Tabellen vor dem Speichern in Qlik Sense auf Basis der Schnittmenge der Tabellendaten reduziert werden. Der ausgeführte Vergleich entspricht einer Verknüpfung von Tabellen über alle gemeinsamen Felder, d. h. in gleicher Weise wie bei einer entsprechenden Verknüpfung. Die Tabellen werden jedoch nicht zusammengeschlossen, sondern als zwei Tabellen unter verschiedenen Namen in Qlik Sense gespeichert.

Syntax:

```
(inner | left | right) keep [(tablename ) ]( loadstatement | selectstatement
)
```

Dem **keep**-Zusatz muss stets einer der Zusätze **inner**, **left** oder **right** vorangehen.

Der explizite Zusatz **join** im Qlik Sense-Skript bewirkt eine vollständige Zusammenfügung der beiden Tabellen, d. h. die beiden Tabellen werden zu einer zusammengeschlossen. Dies führt oft zu umfangreichen und speicherintensiven Tabellen. Einer der großen Vorteile von Qlik Sense besteht aber gerade darin, dass automatisch Verknüpfungen zwischen Tabellen hergestellt werden, ohne dass eine Zusammenfügung erfolgt. Dadurch wird in erheblichem Umfang Speicherplatz gespart, die Zugriffszeiten verkürzen sich und das System behält eine hohe Flexibilität. Aus diesem Grund sollten Sie in Qlik Sense-Skripten im Allgemeinen auf explizite Zusammenfügungen verzichten. Durch den keep-Zusatz reduziert sich zusätzlich die Zahl der Fälle, in denen explizite Zusammenfügungen benötigt werden.

Argumente:

Argumente

Argument	Beschreibung
tablename	Die benannte Tabelle, die mit der geladenen Tabelle verglichen wird.
loadstatementoder selectstatement	Der **LOAD**- oder **SELECT**-Befehl für die geladene Tabelle.

Beispiel:

```
Inner Keep LOAD * from abc.csv;

Left Keep SELECT * from table1;

tab1:

LOAD * from file1.csv;

tab2:

LOAD * from file2.csv;

.. .. ..

Left Keep (tab1) LOAD * from file3.csv;
```

Left

Den Befehlen **Join**und **Keep** kann der Zusatz **left** vorangestellt werden.

Vor einem **join**-Befehl bestimmt dieser Zusatz, dass ein Left Join hergestellt werden soll. Die entstandene Tabelle enthält somit lediglich Kombinationen von Feldwerten der Datentabellen, wobei die Referenzen zu Feldwerten in der ersten Tabelle dargestellt werden. Bei Verwendung vor **keep** bestimmt dieser Zusatz, dass die zweite Rohdatentabelle auf ihre gemeinsame Schnittmenge mit der ersten Tabelle reduziert werden soll, bevor sie in Qlik Sense gespeichert wird.

Haben Sie nach der Stringfunktion mit diesem Namen gesucht? Siehe: Left (page 1491)

Syntax:

```
Left ( Join | Keep) [ (tablename) ](loadstatement | selectstatement)
```

Argumente:

Argumente

Argument	Beschreibung
tablename	Die benannte Tabelle, die mit der geladenen Tabelle verglichen wird.
loadstatementoder selectstatement	Der **LOAD**- oder **SELECT**-Befehl für die geladene Tabelle.

Beispiel

Ladeskript

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Fügen Sie einem Arbeitsblatt in Ihrer App dann die Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um das Ergebnis anzuzeigen.

```
Table1: Load * inline [ Column1, Column2 A, B 1, aa 2, cc 3, ee ];  Table2: Left Join Load *
inline [ Column1, Column3 A, C 1, xx 4, yy ];
```

Ergebnis

Ergebnistabelle

Column1	Column2	Column3
A	B	C
1	aa	xx
2	cc	-
3	ee	-

Erläuterung

Dieses Beispiel zeigt die Ausgabe für den Left Join, wenn nur Werte, die in der ersten (linken) Tabelle vorhanden sind, verknüpft werden.

Mapping

Mithilfe des Zusatzes **mapping** wird eine Mapping-Tabelle erstellt, die z. B. zum Austauschen der Feldwerte und Feldnamen während der Ausführung des Skripts verwendet werden kann.

Syntax:

```
Mapping( loadstatement | selectstatement )
```

Der **mapping**-Zusatz kann vor dem Befehl **LOAD** oder **SELECT** eingefügt werden und speichert das Ergebnis des Load-Befehls als Mapping-Tabelle. Mapping ist eine bequeme Möglichkeit, um Feldwerte während der Skriptausführung zu ersetzen, z. B. zum Ersetzen von US, U.S. oder America durch USA. Eine Mapping-Tabelle besteht aus zwei Spalten: Die erste enthält Vergleichswerte, die zweite die zugehörigen Mapping-Werte. Mapping-Tabellen werden lediglich während der Ausführung des Skripts für das Mapping benötigt und danach wieder gelöscht.

Auf den Inhalt der Mapping-Tabelle kann z. B. mithilfe des Befehls **Map ... Using**, des Befehls **Rename Field**, der Funktion **Applymap()** oder der Funktion **Mapsubstring()** zugegriffen werden.

Beispiel:

In diesem Beispiel wird eine Liste von Verkäufern mit einem Ländercode geladen, der den Wohnsitzstaat enthält. Eine Tabelle, in der die Ländercodes den einzelnen Ländern zugeordnet sind, wird dazu verwendet, den Ländercode durch den tatsächlichen Namen des jeweiligen Landes zu ersetzen. Nur drei Länder sind in Mapping-Tabelle definiert, die anderen Ländercodes sind 'Rest of the world' zugeordnet.

```
// Load mapping table of country codes:
map1:
mapping LOAD *
Inline [
CCode, Country
Sw, Sweden
Dk, Denmark
No, Norway
] ;
// Load list of salesmen, mapping country code to country
// If the country code is not in the mapping table, put Rest of the world
Salespersons:
LOAD *,
ApplyMap('map1', CCode,'Rest of the world') As Country
Inline [
CCode, Salesperson
Sw, John
Sw, Mary

Sw, Per
Dk, Preben
Dk, Olle
No, Ole
Sf, Risttu] ;
// We don't need the CCode anymore
Drop Field 'CCode';
```

Die sich ergebende Tabelle sieht folgendermaßen aus:

Mapping table

Salesperson	Country
John	Sweden
Mary	Sweden
Per	Sweden
Preben	Denmark
Olle	Denmark
Ole	Norway
Risttu	Rest of the world

Merge

Der Zusatz **Merge** kann zu jedem **LOAD**- oder **SELECT**-Befehl im Skript hinzugefügt werden, um anzugeben, dass die geladene Tabelle mit einer anderen Tabelle zusammengeführt werden soll. Er gibt auch an, dass dieser Befehl in einem partiellen Ladevorgang ausgeführt werden soll.

In einem typischen Fall laden Sie ein Änderungsprotokoll und möchten es verwenden, um inserts, updates und deletes auf eine vorhandene Tabelle anzuwenden.

Damit der partielle Ladevorgang korrekt funktioniert, muss die App mit Daten geöffnet werden, bevor der partielle Ladevorgang ausgelöst wird.

Führen Sie einen partiellen Ladevorgang durch, indem Sie die Schaltfläche **Laden** nutzen. Sie können auch das Qlik Engine JSON API verwenden.

Syntax:

```
Merge [only] [(SequenceNoField [, SequenceNoVar])] On ListOfKeys [Concatenate
[(TableName)]] (loadstatement | selectstatement)
```

Argumente:

Argumente

Argument	Beschreibung
only	Ein optionaler Qualifizierer, der bewirkt, dass die Anweisung nur bei der partiellen Ausführung des Skripts berücksichtigt wird. Bei normalen (nicht partiellen) Ladevorgängen wird die Anweisung ignoriert.
SequenceNoField	Der Name des Felds, das einen Zeitstempel oder eine Sequenznummer enthält, die die Vorgangsreihenfolge bestimmt.

Argument	Beschreibung
SequenceNoVar	Der Name der Variablen, der ein Höchstwert für SequenceNoField in der zusammenzuführenden Tabelle zugewiesen wird.
ListOfKeys	Eine kommagetrennte Liste von Feldnamen, die den Primärschlüssel angeben.
Operation	Das erste Feld der load-Anweisung muss die Operation als Textstring enthalten: „Insert“, „Update“ oder „Delete“. „i“, „u“ und „d“ werden ebenfalls akzeptiert.

Allgemeine Funktionen

Während eines normalen (nicht partiellen) Ladevorgangs funktioniert die Konstruktion **Merge LOAD** als normale **load**-Anweisung, aber mit der zusätzlichen Funktion, dass ältere obsolete Datensätze und für die Löschung vorgemerkte Datensätze entfernt werden. Das erste Feld der **load**-Anweisung muss Informationen zum Vorgang enthalten: Insert, Update oder Delete.

Für jeden geladenen Datensatz wird der Datensatzidentifikator mit den zuvor geladenen Datensätzen verglichen, und nur der neueste Datensatz (entsprechend der Sequenznummer) wird beibehalten. Wenn der neueste Datensatz mit Delete markiert ist, wird keiner beibehalten.

Zieltabelle

Anhand des Feldersatzes wird entschieden, welche Tabelle geändert werden muss. Wenn eine Tabelle mit dem gleichen Feldersatz (außer dem ersten Feld, der Operation) bereits vorhanden ist, ist dies die zu ändernde Tabelle. Alternativ kann ein Präfix **Concatenate** verwendet werden, um die Tabelle anzugeben. Wenn die Zieltabelle nicht bestimmt ist, wird das Ergebnis der Konstruktion **Merge LOAD** in einer neuen Tabelle gespeichert.

Wenn das Präfix „Concatenate“ verwendet wird, enthält die Ergebnistabelle einen Satz Felder entsprechend der Verbindung der vorhandenen Tabelle und der Eingabe zur Zusammenführung. Daher kann die Zieltabelle mehr Felder erhalten als das Änderungsprotokoll, das als Eingabe für die Zusammenführung verwendet wird.

Ein partieller Ladevorgang hat die gleiche Wirkung wie ein vollständiger Ladevorgang. Ein Unterschied besteht darin, dass ein partieller Ladevorgang nur selten eine neue Tabelle erstellt. Außer bei Verwendung der Bedingung **Only** ist immer bereits eine Zieltabelle mit dem gleichen Feldersatz aus der vorherigen Skriptausführung vorhanden.

Sequenznummer

Wenn es sich bei dem geladenen Änderungsprotokoll um ein kumuliertes Protokoll handelt, das bereits geladene Änderungen enthält, kann der Parameter SequenceNoVar in einem **Where**-Befehl verwendet werden, um die Menge der Eingabedaten zu begrenzen. **Merge LOAD** kann dann nur zum Laden von Datensätzen angewendet werden, in denen das Feld SequenceNoField größer als SequenceNoVar ist. Nach Abschluss weist **Merge LOAD** einen neuen Wert zu SequenceNoVar zu, wobei der Wert im Feld SequenceNoField der Höchstwert ist.

Operationen

Merge LOAD kann weniger Felder als die Zieltabelle enthalten. Die verschiedenen Operationen behandeln fehlende Felder unterschiedlich:

Einfügen: Felder, die in **Merge LOAD** fehlen, aber in der Zieltabelle vorhanden sind, erhalten eine NULL in der Zieltabelle.

Löschen: Fehlende Felder wirken sich nicht auf das Ergebnis aus. Die betreffenden Datensätze werden dennoch gelöscht.

Aktualisieren: In **Merge LOAD** aufgelistete Felder werden in der Zieltabelle aktualisiert. Fehlende Felder werden nicht geändert. Das bedeutet, dass die beiden folgenden Anweisungen nicht identisch sind:

- Merge on Key Concatenate Load 'U' as Operation, Key, F1, Null() as F2 From ...;
- Merge on Key Concatenate Load 'U' as Operation, Key, F1 From ...;

Mit der ersten Anweisung werden die aufgelisteten Datensätze aktualisiert und F2 zu NULL geändert. Mit dem zweiten wird F2 nicht geändert, sondern die Werte werden in der Zieltabelle abgelegt.

Beispiele

Beispiel 1: Einfaches Zusammenführen mit der angegebenen Tabelle

In diesem Beispiel wird eine Inline-Tabelle mit dem Namen Persons mit drei Zeilen geladen. **Merge** ändert dann die Tabelle wie folgt:

- Fügt die Zeile *Mary, 4* hinzu.
- Löscht die Zeile *Steven, 3*.
- Weist die Nummer *5* zu *Jake* zu.

Die Variable *LastChangeDate* wird auf den Höchstwert in der Spalte *ChangeDate* festgelegt, nachdem **Merge** ausgeführt wurde.

Ladeskript

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Fügen Sie einem Arbeitsblatt in Ihrer App dann die Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um das Ergebnis anzuzeigen.

```
Set DateFormat='D/M/YYYY';
Persons:
load * inline [
Name, Number
Jake, 3
Jill, 2
Steven, 3
];

Merge (ChangeDate, LastChangeDate) on Name Concatenate(Persons)
LOAD * inline [
Operation, ChangeDate,   Name,     Number
Insert,    1/1/2021,     Mary,     4
Delete,    1/1/2021,     Steven,
```

```
Update,    2/1/2021,    Jake,    5
];
```

Ergebnis

Vor dem **Merge Load** sieht die Ergebnistabelle wie folgt aus:

Resulting table

Name	**Number**
Jake	3
Jill	2
Steven	3

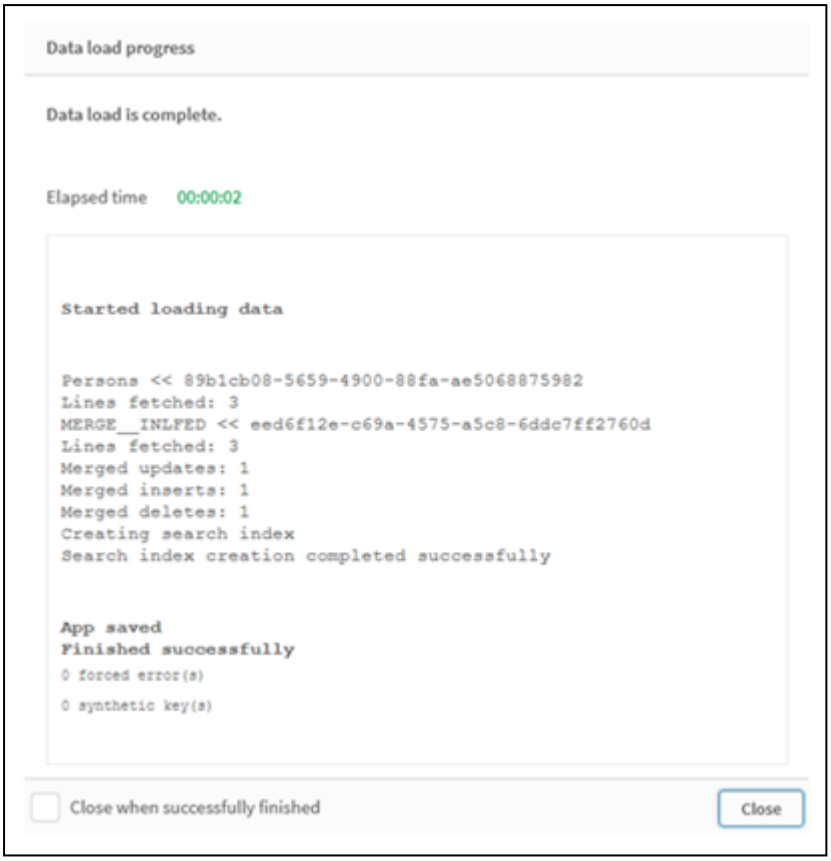
Nach dem **Merge Load** sieht die Ergebnistabelle wie folgt aus:

Resulting table

ChangeDate	**Name**	**Number**
2/1/2021	Jake	5
-	Jill	2
1/1/2021	Mary	4

Wenn die Daten geladen werden, zeigt das Dialogfeld **Datenladefortschritt** die durchgeführten Vorgänge an:

Dialogfeld „Datenladefortschritt“



Beispiel 2: Datenladeskript mit fehlenden Feldern

In diesem Beispiel werden die gleichen Daten wie oben geladen, aber jetzt mit einer ID pro Person.

Merge ändert die Tabelle wie folgt:

- Fügt die Zeile *Mary, 4* hinzu.
- Löscht die Zeile *Steven, 3*.
- Weist die Nummer *5* zu *Jake* zu.
- Weist die Nummer *6* zu *Jill* zu.

Ladeskript

Hier verwenden wir zwei **Merge Load**-Anweisungen, eine für das „Einfügen“ und „Löschen“ und eine für „Aktualisieren“.

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Fügen Sie einem Arbeitsblatt in Ihrer App dann die Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um das Ergebnis anzuzeigen.

```
Set DateFormat='D/M/YYYY';
Persons:
Load * Inline [
PersonID, Name, Number
1, Jake, 3
2, Jill, 2
3, Steven, 3
```

```
];

Merge (ChangeDate, LastChangeDate) on PersonID Concatenate(Persons)
Load * Inline [
Operation, ChangeDate,   PersonID, Name,     Number
Insert,    1/1/2021,     4,        Mary,     4
Delete,    1/1/2021,     3,        Steven,
];

Merge (ChangeDate, LastChangeDate) on PersonID Concatenate(Persons)
Load * Inline [
Operation, ChangeDate,   PersonID, Number
Update,    2/1/2021,     1,        5
Update,    3/1/2021,     2,        6
];
```

Ergebnis

Nach den **Merge Load**-Anweisungen sieht die Tabelle wie folgt aus:

Resulting table

PersonID	ChangeDate	Name	Number
1	2/1/2021	Jake	5
2	3/1/2021	Jill	6
4	1/1/2021	Mary	4

Beachten Sie, dass die zweite **Merge**-Anweisung nicht das Feld **Name** enthält. Daher werden die Namen nicht geändert.

Beispiel 3: Datenladeskript – partieller Ladevorgang mit einer Where-Bedingung mit ChangeDate

Im folgenden Beispiel gibt das Argument **Only** an, dass der Befehl **Merge** nur während eines partiellen Ladevorgangs ausgeführt wird. Aktualisierungen werden gestützt auf das zuvor erfasste „LastChangeDate“ gefiltert. Nachdem **Merge** abgeschlossen ist, wird die Variable „LastChangeDate“ dem Höchstwert der Spalte „ChangeDate“ zugewiesen, die während der Zusammenführung verarbeitet wurde.

Ladeskript

```
Merge Only (ChangeDate, LastChangeDate) on Name Concatenate(Persons)
LOAD Operation, ChangeDate,  Name, Number
from [lib://ChangeFilesFolder/BulkChangesInPersonsTable.csv] (txt)
where ChangeDate >='$(LastChangeDate)';
```

NoConcatenate

Der Zusatz **NoConcatenate** bewirkt, dass zwei geladene Tabellen mit identischen Feldergruppen als zwei separate interne Tabellen behandelt werden, während sie andernfalls automatisch zusammengefasst werden würden.

Syntax:

```
NoConcatenate( loadstatement | selectstatement )
```

Standardmäßig fasst Qlik Sense zwei Tabellen automatisch zusammen, wenn eine Tabelle geladen wird, die eine identische Anzahl Felder und übereinstimmende Feldnamen wie eine andere Tabelle aufweist, die zuvor in das Skript geladen wurde. Dies trifft selbst dann zu, wenn die zweite Tabelle einen anderen Namen hat.

Wenn jedoch das Skriptpräfix `NoConcatenate` vor die load-Anweisung oder die select-Anweisung der zweiten Tabelle gesetzt wird, dann werden diese beiden Tabellen getrennt geladen.

Ein typischer Anwendungsfall für `NoConcatenate` ist, wenn Sie eine temporäre Kopie einer Tabelle erstellen möchten, um temporäre Umwandlungen in dieser Kopie vorzunehmen und eine Kopie der Originaldaten beizubehalten. `NoConcatenate` sorgt dafür, dass Sie diese Kopie erstellen können, ohne sie implizit wieder der Quelltabelle hinzuzufügen.

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Funktionsbeispiel

Beispiel	**Ergebnis**
`Source: LOAD A,B from file1.csv; CopyOfSource: NoConcatenate LOAD A,B resident Source;`	Eine Tabelle mit `A` und `B` als Kennzahlen wird geladen. Eine zweite Tabelle mit den gleichen Feldern wird getrennt geladen, weil die Variable `NoConcatenate` verwendet wird.

Beispiel 1 – Implizite Zusammenfassung

Ladeskript und Ergebnisse

Übersicht

In diesem Beispiel fügen Sie zwei Ladeskripte in Reihenfolge hinzu.

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein anfänglicher Datensatz mit Datumswerten und Beträgen wird an eine Tabelle namens `Transactions` gesendet.

Erstes Ladeskript

```
Transactions:
LOAD
*
Inline [
id, date, amount
1, 08/30/2018, 23.56
2, 09/07/2018, 556.31
3, 09/16/2018, 5.75
4, 09/22/2018, 125.00
5, 09/22/2018, 484.21
6, 09/22/2018, 59.18
7, 09/23/2018, 177.42
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `id`
- `date`
- `amount`

Erste Ergebnistabelle

id	date	amount
1	08/30/2018	23.56
2	09/07/2018	556.31
3	09/16/2018	5.75
4	09/22/2018	125.00
5	09/22/2018	484.21
6	09/22/2018	59.18
7	09/23/2018	177.42

Zweites Ladeskript

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein zweiter Datensatz mit identischen Feldern wird an eine Tabelle namens `Sales` gesendet.

```
Sales:
LOAD
```

```
*
Inline [
id, date, amount
8, 10/01/2018, 164.27
9, 10/03/2018, 384.00
10, 10/06/2018, 25.82
11, 10/09/2018, 312.00
12, 10/15/2018, 4.56
13, 10/16/2018, 90.24
14, 10/18/2018, 19.32
];
```

Ergebnisse

Laden Sie die Daten und wechseln Sie zur Tabelle.

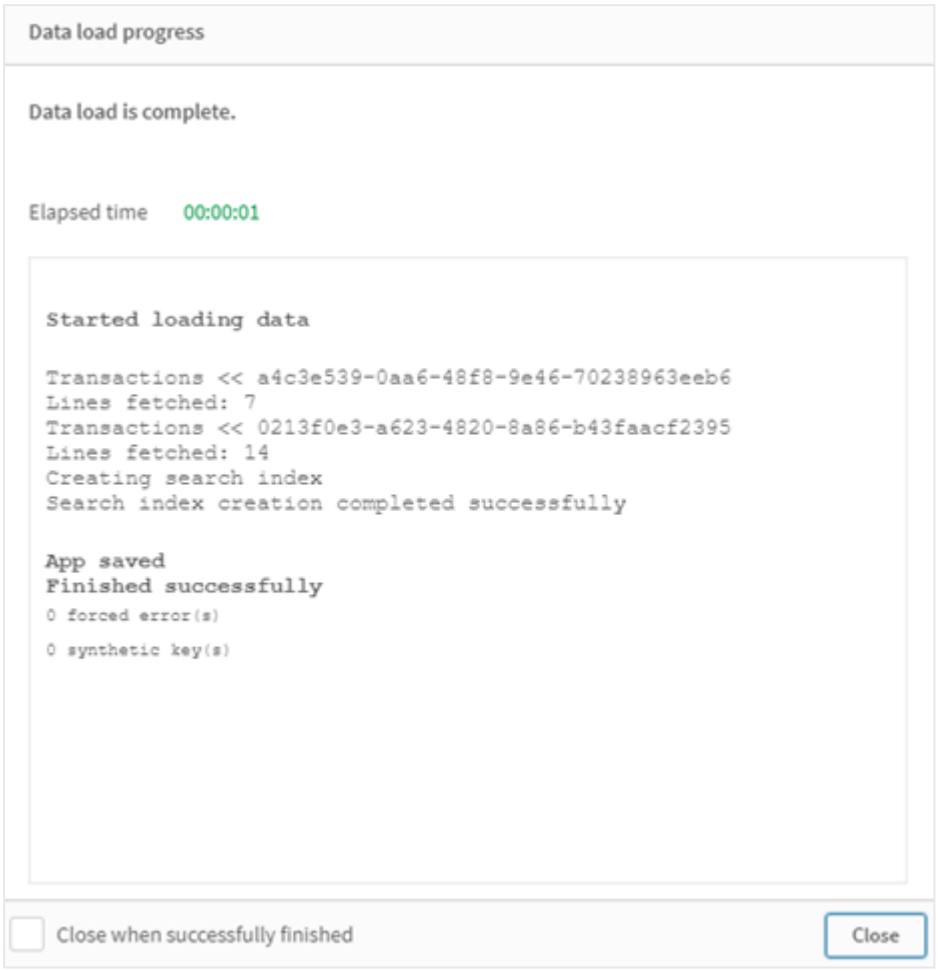
Zweite Ergebnistabelle

id	date	amount
1	08/30/2018	23.56
2	09/07/2018	556.31
3	09/16/2018	5.75
4	09/22/2018	125.00
5	09/22/2018	484.21
6	09/22/2018	59.18
7	09/23/2018	177.42
8	10/01/2018	164.27
9	10/03/2018	384.00
10	10/06/2018	25.82
11	10/09/2018	312.00
12	10/15/2018	4.56
13	10/16/2018	90.24
14	10/18/2018	19.32

Wenn das Skript ausgeführt wird, wird die Tabelle `Sales` implizit mit der vorhandenen Tabelle `Transactions` zusammengefasst, da die beiden Datensätze die gleiche Anzahl Felder mit identischen Namen haben. Dies geschieht, obwohl versucht wird, den Ergebnissatz mit einem Namens-Tag für die zweite Tabelle `'Sales'` zu nennen.

Im Protokoll **Datenladefortschritt** können Sie sehen, dass der Datensatz „Sales“ implizit zusammengefasst wurde.

Das Protokoll „Datenladefortschritt“ zeigt Transaktionsdaten, die implizit zusammengefasst wurden.



Beispiel 2 –Anwendungsfall-Szenario

Ladeskript und Ergebnisse

Übersicht

Dieses Anwendungsfall-Szenario enthält:

- Einen Transaktionsdatensatz mit:
    - id
    - date
    - Betrag (in GBP)
- Eine Währungstabelle mit:
    - Wechselkursraten für USD in GBP
- Einen zweiten Transaktionsdatensatz mit:
    - id

- date
- Betrag (in USD)

Sie laden fünf Skripte in Reihenfolge.

- Das erste Ladeskript enthält einen anfänglichen Datensatz mit Datumswerten und Beträgen in GBP, der an eine Tabelle namens `Transactions` gesendet wird.
- Das zweite Ladeskript enthält:
    - Ein zweiter Datensatz mit Datumswerten und Beträgen in USD wird an eine Tabelle namens `Transactions_in_USD` gesendet.
    - Das Präfix `noconcatenate` wird vor der load-Anweisung des Datensatzes `Transactions_in_USD` platziert, um eine implizite Zusammenfassung zu vermeiden.
- Das dritte Ladeskript enthält das Präfix `join`, das verwendet wird, um eine Wechselkursrate zwischen GBP und USD in der Tabelle `Transactions_in_USD` zu erstellen.
- Das vierte Ladeskript enthält das Präfix `concatenate`, mit dem die `Transactions_in_USD` zur anfänglichen Tabelle `Transactions` hinzugefügt werden.
- Das fünfte Ladeskript enthält den Befehl `drop table`, mit dem die Tabelle `Transactions_in_USD` entfernt wird, nachdem ihre Daten mit der Tabelle `Transactions` zusammengefasst wurden.

Erstes Ladeskript

```
Transactions:

Load * Inline [
id, date, amount
1, 12/30/2018, 23.56
2, 12/07/2018, 556.31
3, 12/16/2018, 5.75
4, 12/22/2018, 125.00
5, 12/22/2018, 484.21
6, 12/22/2018, 59.18
7, 12/23/2018, 177.42
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `id`
- `date`
- `amount`

Ergebnisse des ersten Ladeskripts

id	date	amount
1	12/30/2018	23.56

id	date	amount
2	12/07/2018	556.31
3	12/16/2018	5.75
4	12/22/2018	125.00
5	12/22/2018	484.21
6	12/22/2018	59.18
7	12/23/2018	177.42

Die Tabelle zeigt den anfänglichen Datensatz mit Beträgen in GBP.

Zweites Ladeskript

```
Transactions_in_USD:
NoConcatenate
Load * Inline [
id, date, amount
8, 01/01/2019, 164.27
9, 01/03/2019, 384.00
10, 01/06/2019, 25.82
11, 01/09/2019, 312.00
12, 01/15/2019, 4.56
13, 01/16/2019, 90.24
14, 01/18/2019, 19.32
];
```

Ergebnisse

Laden Sie die Daten und wechseln Sie zur Tabelle.

Ergebnisse des zweiten Ladeskripts

id	date	amount
1	12/30/2018	23.56
2	12/07/2018	556.31
3	12/16/2018	5.75
4	12/22/2018	125.00
5	12/22/2018	484.21
6	12/22/2018	59.18
7	12/23/2018	177.42
8	01/01/2019	164.27
9	01/03/2019	384.00
10	01/06/2019	25.82

id	date	amount
11	01/09/2019	312.00
12	01/15/2019	4.56
13	01/16/2019	90.24
14	01/18/2019	19.32

Sie sehen, dass der zweite Datensatz aus der Tabelle `Transactions_in_USD` hinzugefügt wurde.

Drittes Ladeskript

Dieses Ladeskript verknüpft eine Wechselkursrate von USD zu GBP mit der Tabelle `Transactions_in_USD`.

```
Join (Transactions_in_USD)
Load * Inline [
rate
0.7
];
```

Ergebnisse

Laden Sie die Daten und gehen Sie zur Datenmodellansicht. Wählen Sie die Tabelle `Transactions_in_USD` aus. Sie sehen, dass jeder vorhandene Datensatz einen Feldwert für „rate“ von 0,7 enthält.

Viertes Ladeskript

Mithilfe von Resident Load fasst dieses Ladeskript die Tabelle `Transactions_in_USD` mit der Tabelle `Transactions` zusammen, nachdem die Beträge in USD umgerechnet wurden.

```
Concatenate (Transactions)
LOAD
id,
date,
amount * rate as amount
Resident Transactions_in_USD;
```

Ergebnisse

Laden Sie die Daten und wechseln Sie zur Tabelle. Sie sehen neue Einträge mit Beträgen in GBP von Zeile acht bis vierzehn.

Ergebnisse des vierten Ladeskripts

id	date	amount
1	12/30/2018	23.56
2	12/07/2018	556.31
3	12/16/2018	5.75

id	date	amount
4	12/22/2018	125.00
5	12/22/2018	484.21
6	12/22/2018	59.18
7	12/23/2018	177.42
8	01/01/2019	114.989
8	01/01/2019	164.27
9	01/03/2019	268.80
9	01/03/2019	384.00
10	01/06/2019	18.074
10	01/06/2019	25.82
11	01/09/2019	218.40
11	01/09/2019	312.00
12	01/15/2019	3.192
12	01/15/2019	4.56
13	01/16/2019	63.168
13	01/16/2019	90.24
14	01/18/2019	13.524
14	01/18/2019	19.32

Fünftes Ladeskript

Dieses Ladeskript entfernt die duplizierten Einträge aus der Ergebnistabelle des vierten Ladeskripts und belässt nur Einträge mit Beträgen in GBP.

```
drop tables Transactions_in_USD;
```

Ergebnisse

Laden Sie die Daten und wechseln Sie zur Tabelle.

Ergebnisse des fünften Ladeskripts

id	date	amount
1	12/30/2018	23.56
2	12/07/2018	556.31
3	12/16/2018	5.75
4	12/22/2018	125.00

id	date	amount
5	12/22/2018	484.21
6	12/22/2018	59.18
7	12/23/2018	177.42
8	01/01/2019	114.989
9	01/03/2019	268.80
10	01/06/2019	18.074
11	01/09/2019	218.40
12	01/15/2019	3.192
13	01/16/2019	63.168
14	01/18/2019	13.524

Nach dem Laden des fünften Ladeskripts zeigt die Ergebnistabelle alle vierzehn Transaktionen, die in beiden Transaktionsdatensätzen vorhanden waren; für Transaktion 8-14 wurden jedoch die Beträge in GBP umgerechnet.

Wenn das Präfix `NoConcatenate` entfernt wird, das im zweiten Ladeskript vor `Transactions_in_USD` verwendet wurde, schlägt das Skript mit dem Fehler „Tabelle `Transactions_in_USD` nicht gefunden“ fehl. Dies liegt daran, dass die Tabelle `Transactions_in_USD` automatisch mit der Originaltabelle `Transactions` zusammengefasst worden wäre.

Only

Das Skriptschlüsselwort **Only** wird als Aggregierungsfunktion oder als Teil der Syntax in den Präfixen **Add**, **Replace** und **Merge** bei der partiellen Ausführung des Skripts verwendet.

Outer

Dem expliziten Präfix **Join** kann das Präfix **Outer** vorangestellt werden, um einen Outer Join festzulegen. In einem Outer Join werden alle Kombinationen zwischen den zwei Tabellen erzeugt. Die entstandene Tabelle enthält somit Kombinationen von Feldwerten der Rohdatentabellen, wobei die verknüpfenden Feldwerte in beiden Tabellen dargestellt werden. Das Schlüsselwort **Outer** ist optional und ist der standardmäßige Typ, der verwendet wird, wenn kein Join-Präfix festgelegt wird.

Syntax:

```
Outer Join [ (tablename) ](loadstatement |selectstatement )
```

Argumente:

Argumente

Argument	Beschreibung
tablename	Die benannte Tabelle, die mit der geladenen Tabelle verglichen wird.
loadstatementoder selectstatement	Der **LOAD**- oder **SELECT**-Befehl für die geladene Tabelle.

Beispiel

Ladeskript

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Fügen Sie einem Arbeitsblatt in Ihrer App dann die Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um das Ergebnis anzuzeigen.

```
Table1: Load * inline [ Column1, Column2 A, B 1, aa 2, cc 3, ee ];  Table2: Outer Join Load *
inline [ Column1, Column3 A, C 1, xx 4, yy ];
```

Ergebnistabelle

Column1	**Column2**	**Column3**
A	B	C
1	aa	xx
2	cc	-
3	ee	-
4	-	yy

Erläuterung

In diesem Beispiel werden die beiden Tabellen Table1 und Table2 zu einer einzigen Tabelle namens Table1 zusammengeführt. In solchen Fällen wird häufig das Präfix **outer** verwendet, um mehrere Tabelle in einer einzigen Tabelle zu verknüpfen und Aggregierungen anhand der Werte einer einzelnen Tabelle durchzuführen.

Partielles Laden

Ein vollständiger Ladevorgang beginnt immer mit dem Löschen aller Tabellen im vorhandenen Datenmodell. Dann wird das Ladeskript ausgeführt.

Ein partieller Ladevorgang tut dies nicht. Stattdessen werden alle Tabellen im Datenmodell beibehalten und dann nur die Anweisungen **Load** und **Select** mit den Zusätzen **Add**, **Merge** oder **Replace** ausgeführt. Andere Datentabellen sind von dem Befehl nicht betroffen. Das Argument **only** gibt an, dass die Anweisung nur bei partiellen Ladevorgängen ausgeführt und bei vollständigen Ladevorgängen ignoriert werden soll. Die folgende Tabelle bietet eine Zusammenfassung der Anweisungsausführung für teilweise und vollständige Ladevorgänge.

Anweisung	**Vollständiges Laden**	**Partielles Laden**
Load ...	Anweisung wird ausgeführt	Anweisung wird nicht ausgeführt
Add/Replace/Merge Load ...	Anweisung wird ausgeführt	Anweisung wird ausgeführt

Anweisung	Vollständiges Laden	Partielles Laden
Add/Replace/Merge Only Load ...	Anweisung wird nicht ausgeführt	Anweisung wird ausgeführt

Partielle Ladevorgänge haben im Vergleich zu vollständigen Ladevorgängen mehrere Vorteile:

- Sie sind schneller, da nur die kürzlich geänderten Daten geladen werden müssen. Bei großen Datensätzen ist der Unterschied signifikant.
- Es wird weniger Speicher verbraucht, da weniger Daten geladen werden.
- Der Vorgang ist zuverlässiger, da Abfragen an Datenquellen schneller ausgeführt werden, was das Risiko von Netzwerkproblemen reduziert.

Damit der partielle Ladevorgang korrekt funktioniert, muss die App mit Daten geöffnet werden, bevor der partielle Ladevorgang ausgelöst wird.

Führen Sie einen partiellen Ladevorgang durch, indem Sie die Schaltfläche **Laden** nutzen. Sie können auch das Qlik Engine JSON API verwenden.

Einschränkungen

Ein teilweises Laden schlägt fehl, wenn Befehle mit Referenzen auf Tabellen vorhanden sind, die während des vollständigen Ladens vorhanden waren, aber nicht während des teilweisen Ladens.

Beispiel

Beispielbefehle

```
LEFT JOIN(<Table_removed_after_full_reload>)
CONCATENATE(<Table_removed_after_full_reload>)
```

Dabei ist <Table_removed_after_full_reload> eine Tabelle, die beim vollständigen Laden vorhanden war, aber nicht beim teilweisen Laden.

Problemumgehung

Als Problemumgehung können Sie den Befehl mit dem folgenden if-Befehl umschließen.

```
IF NOT IsPartialReload() THEN ... ENDIF.
```

Ein partieller Ladevorgang kann Werte aus den Daten entfernen. Dies wird jedoch nicht in der Liste der distinkten Werte widergespiegelt, die eine intern gepflegte Tabelle ist. Nach einem partiellen Ladevorgang enthält die Liste daher alle distinkten Werte, die im Feld seit dem letzten vollständigen Ladevorgang vorhanden waren, was mehr sein können, als derzeit nach dem partiellen Ladevorgang vorhanden sind. Dies wirkt sich auf die Ausgabe der Funktionen FieldValueCount() und FieldValue() aus. FieldValueCount() könnte eine größere Zahl als die aktuelle Zahl der Feldwerte zurückgeben.

Beispiel

Beispiel 1

Ladeskript

Fügen Sie Ihrer App das Beispielskript hinzu und führen Sie einen partiellen Ladevorgang aus. Fügen Sie einem Arbeitsblatt in Ihrer App dann die Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um das Ergebnis anzuzeigen.

```
T1:
Add only Load distinct recno()+10 as Num autogenerate 10;
```

Ergebnis

Resulting table

Num	**Count(Num)**
11	1
12	1
13	1
14	1
15	1
16	1
17	1
18	1
19	1
20	1

Erläuterung

Die Anweisung wird nur während eines partiellen Ladevorgangs ausgeführt. Wenn das Präfix „distinct“ weggelassen wird, erhöht sich die Anzahl des Felds **Num** mit jedem nachfolgenden partiellen Ladevorgang.

Beispiel 2

Ladeskript

Fügen Sie der App das Beispielskript hinzu. Führen Sie einen vollständigen Ladevorgang aus und zeigen Sie das Ergebnis an. Führen Sie dann einen partiellen Ladevorgang aus und zeigen Sie das Ergebnis an. Fügen Sie einem Arbeitsblatt in Ihrer App dann die Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um die Ergebnisse anzuzeigen.

```
T1:
Load recno() as ID, recno() as Value autogenerate 10;

T1:
Replace only Load recno() as ID, repeat(recno(),3) as Value autogenerate 10;
```

Ergebnis

Output table after full reload

ID	Value
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10

Output table after partial reload

ID	Value
1	111
2	222
3	333
4	444
5	555
6	666
7	777
8	888
9	999
10	101010

Erläuterung

Die erste Tabelle wird während eines vollständigen Ladevorgangs geladen, und die zweite Tabelle ersetzt einfach die erste Tabelle während eines partiellen Ladevorgangs.

Replace

Das Skriptschlüsselwort **Replace** wird als Skriptfunktion oder bei einer partiellen Ausführung des Skripts als Zusatz verwendet.

Replace

Der Zusatz **Replace** kann zu jedem **LOAD**- oder **SELECT**-Befehl im Skript hinzugefügt werden, um anzugeben, dass die geladene Tabelle eine andere Tabelle ersetzen soll. Er gibt auch an, dass dieser Befehl in einem partiellen Ladevorgang ausgeführt werden soll. Der Zusatz **Replace** kann auch in einem **Map**-Befehl verwendet werden.

> *Damit der partielle Ladevorgang korrekt funktioniert, muss die App mit Daten geöffnet werden, bevor der partielle Ladevorgang ausgelöst wird.*

Führen Sie einen partiellen Ladevorgang durch, indem Sie die Schaltfläche **Laden** nutzen. Sie können auch das Qlik Engine JSON API verwenden.

Syntax:

```
Replace [only] [Concatenate[(tablename)]] (loadstatement | selectstatement)
```

```
Replace [only] mapstatement
```

Während eines normalen (nicht partiellen) Ladevorgangs funktioniert die Konstruktion **Replace LOAD** wie ein normaler **LOAD**-Befehl, dem aber **Drop Table** vorangestellt wird. Zuerst wird die alte Tabelle gelöscht, dann werden Datensätze generiert und als neue Tabelle gespeichert.

Wenn der Zusatz **Concatenate** verwendet wird oder wenn eine Tabelle mit dem gleichen Satz Felder vorhanden ist, dann ist dies die zu löschende Tabelle. Wenn keine zu löschende Tabelle vorhanden ist, verhält sich die Konstruktion **Replace LOAD** genau wie ein normaler **LOAD**.

Ein partieller Ladevorgang hat die gleiche Wirkung. Der einzige Unterschied besteht darin, dass aus der vorherigen Skriptausführung immer eine zu löschende Tabelle vorhanden ist. Die Konstruktion **Replace LOAD** löscht immer zuerst die alte Tabelle und erstellt dann eine neue.

Durch den **Replace Map...Using**-Befehl wird auch bei der partiellen Ausführung des Skripts ein Mapping durchgeführt.

Argumente:

Argumente

Argument	Beschreibung
only	Ein optionaler Qualifizierer, der bewirkt, dass der Befehl nur bei der partiellen Ausführung des Skripts berücksichtigt wird. Bei normalen (nicht partiellen) Ladevorgängen sollte er ignoriert werden.

Beispiele und Ergebnisse:

Beispiel	Ergebnis
`Tab1: Replace LOAD * from File1.csv;`	Bei der normalen sowie bei der partiellen Ausführung des Skripts wird die Qlik Sense-Tabelle Tab1 zunächst gelöscht. Danach werden neue Daten aus der Datei File1.csv geladen und in Tab1 gespeichert.
`Tab1: Replace only LOAD * from File1.csv;`	Bei der normalen Ausführung des Skripts wird dieser Befehl nicht berücksichtigt. Bei der partiellen Ausführung des Skripts wird jede Qlik Sense-Tabelle mit der Bezeichnung Tab1 zunächst gelöscht. Danach werden neue Daten aus der Datei File1.csv geladen und in Tab1 gespeichert.
`Tab1: LOAD a,b,c from File1.csv; Replace LOAD a,b,c from File2.csv;`	Bei der normalen Ausführung des Skripts werden die Daten aus der Datei File1.csv zunächst in die Qlik Sense-Tabelle Tab1 eingelesen, aber sofort wieder gelöscht und durch die neuen Daten aus der Datei File2.csv ersetzt. Sämtliche Daten aus der Datei File1.csv gehen verloren. Bei der partiellen Ausführung des Skripts wird die gesamte Qlik Sense-Tabelle Tab1 zunächst gelöscht. Danach wird sie durch die neuen Daten aus der Datei File2.csv ersetzt.
`Tab1: LOAD a,b,c from File1.csv; Replace only LOAD a,b,c from File2.csv;`	Bei der normalen Ausführung des Skripts werden die Daten aus der Datei File1.csv geladen und in der Qlik Sense-Tabelle Tab1 gespeichert. Die Datei File2.csv wird nicht berücksichtigt. Bei der partiellen Ausführung des Skripts wird die gesamte Qlik Sense-Tabelle Tab1 zunächst gelöscht. Danach wird sie durch die neuen Daten aus der Datei File2.csv ersetzt. Sämtliche Daten aus der Datei File1.csv gehen verloren.

Right

Den Befehlen **Join**und **Keep** kann der Zusatz **right** vorangestellt werden.

Vor einem **join**-Befehl bestimmt dieser Zusatz, dass ein Right Join hergestellt werden soll. Die entstandene Tabelle enthält somit lediglich Kombinationen von Feldwerten der Datentabellen, wobei die Referenzen zu Feldwerten in der zweiten Tabelle dargestellt werden. Bei Verwendung vor **keep** bestimmt dieser Zusatz, dass die erste Rohdatentabelle auf ihre gemeinsame Schnittmenge mit der zweiten Tabelle reduziert werden sollte, bevor sie in Qlik Sense gespeichert wird.

Haben Sie nach der Stringfunktion mit diesem Namen gesucht? Siehe: Right (page 1499)

Syntax:

```
Right (Join | Keep) [(tablename)](loadstatement |selectstatement )
```

Argumente:

Argumente

Argument	Beschreibung
tablename	Die benannte Tabelle, die mit der geladenen Tabelle verglichen wird.
loadstatementoder selectstatement	Der **LOAD**- oder **SELECT**-Befehl für die geladene Tabelle.

Beispiel

Ladeskript

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Fügen Sie einem Arbeitsblatt in Ihrer App dann die Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um das Ergebnis anzuzeigen.

```
Table1: Load * inline [ Column1, Column2 A, B 1, aa 2, cc 3, ee ];  Table2: Right Join Load * inline [ Column1, Column3 A, C 1, xx 4, yy ];
```

Ergebnis

Ergebnistabelle

Column1	Column2	Column3
A	B	C
1	aa	xx
4	-	yy

Erläuterung

Dieses Beispiel zeigt die Ausgabe für den Right Join, wo nur Werte, die in der zweiten (rechten) Tabelle vorhanden sind, verknüpft werden.

Sample

Der Zusatz **sample** vor einem **LOAD**- oder **SELECT**-Befehl dient dazu, eine zufällige Stichprobe von Datensätzen aus einer Datenquelle zu laden.

Syntax:

```
Sample  p ( loadstatement | selectstatement )
```

Die Formel, die ausgewertet wird, definiert nicht den Prozentsatz der Einträge im Datensatz, die in die Qlik Sense-Anwendung geladen werden, sondern die Wahrscheinlichkeit, mit der jeder gelesene Eintrag in die Anwendung geladen wird. Wenn ein Wert von `p = 0.5` angegeben wird, bedeutet dies also nicht, dass 50 % der gesamten Einträge geladen werden, sondern dass jeder Eintrag eine Wahrscheinlichkeit von 50 % hat, in die Qlik Sense-Anwendung geladen zu werden.

Argumente

Argument	Beschreibung
p	Eine beliebige Formel, die eine Zahl größer als 0 und kleiner oder gleich 1 liefert. Die Zahl gibt die Wahrscheinlichkeit an, dass ein bestimmter Datensatz gelesen wird. Es werden alle Datensätze gelesen, jedoch werden nur einige in Qlik Sense geladen.

Verwendung

Stichproben sind nützlich, wenn Sie Beispieldaten aus einer großen Tabelle verwenden möchten, um die Art der Daten, die Verteilung oder den Feldinhalt zu verstehen. Da ein Teilsatz der Daten vorgelegt wird, erfolgen Datenladevorgänge schneller und Skripte können schneller getestet werden. Anders als `First` ruft die Funktion `Sample` Daten aus der ganzen Tabelle ab, anstatt sich auf wenige erste Zeilen zu beschränken. Das kann in manchen Fällen eine genauere Darstellung der Daten bieten.

Die folgenden Beispiele zeigen zwei mögliche Nutzungen des Systempräfix `Sample`:

```
Sample 0.15 SQL SELECT * from Longtable;

Sample(0.15) LOAD * from Longtab.csv;
```

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Beispiel 1 – Stichprobe aus einer Inline-Tabelle

Ladeskript und Ergebnisse

Übersicht

In diesem Beispiel lädt das Skript eine Datenstichprobe aus einem Datensatz mit sieben Einträgen aus einer Inline-Tabelle in eine Tabelle namens `Transactions`.

Ladeskript

```
Transactions:
SAMPLE 0.3
LOAD
*
Inline [
id, date, amount
1, 08/30/2018, 23.56
2, 09/07/2018, 556.31
3, 09/16/2018, 5.75
4, 09/22/2018, 125.00
5, 09/22/2018, 484.21
6, 09/22/2018, 59.18
7, 09/23/2018, 177.42
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `id`
- `amount`

Fügen Sie die folgende Kennzahl hinzu:

```
=sum(amount)8
```

Ergebnistabelle

id	date	=Sum(amount)
2	09/07/2018	556.31
4	09/22/2018	125
1	08/30/2018	23.56
3	09/16/2018	5.75

In der Iteration des in diesem Beispiel verwendeten Ladevorgangs wurden alle sieben Einträge gelesen, aber nur vier Einträge wurden in die Datentabelle geladen. Jeder wiederholte Ladevorgang kann eine andere Zahl ergeben und dazu führen, dass ein anderer Satz Einträge in die Anwendung geladen wird.

Beispiel 2 – Stichprobe einer automatisch generierten Tabelle

Ladeskript und Ergebnisse

Übersicht

In diesem Beispiel, das `Autogenerate` verwendet, wird ein Datensatz aus 100 Einträgen erstellt, der die Felder `date`, `id` und `amount` enthält. Es wird aber das Präfix `Sample` mit einem Wert von 0.1 verwendet.

Ladeskript

```
SampleData:
Sample 0.1
LOAD
RecNo() AS id,
MakeDate(2013, Ceil(Rand() * 12), Ceil(Rand() * 29)) as date,
Rand() * 1000 AS amount

Autogenerate(100);
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `id`
- `amount`

Fügen Sie die folgende Kennzahl hinzu:

Ergebnistabelle

id	date	=Sum(amount)
48	9/28/2013	763
20	5/15/2013	752
19	11/8/2013	657
25	3/24/2013	522
27	8/23/2013	389
81	6/1/2013	53
100	8/15/2013	17

In der in diesem Beispiel verwendeten Iteration des Ladevorgangs wurden sieben Einträge aus dem erstellten Datensatz geladen. Auch hier kann jeder wiederholte Ladevorgang eine andere Zahl ergeben und dazu führen, dass ein anderer Satz Einträge in die Anwendung geladen wird.

Semantic

Der Ladezusatz `semantic` erstellt einen besonderen Feldtyp, der in Qlik Sense verwendet werden kann, um relationale Daten zu verbinden und zu verwalten, beispielsweise Baumstrukturen, selbstreferenzierende Daten in einer über- und untergeordneten Struktur oder Daten, die als Diagramm beschrieben werden können.

Beachten Sie, dass der Ladevorgang `semantic` ähnlich wie die Zusätze *Hierarchy (page 66)* und *HierarchyBelongsTo (page 68)* funktionieren kann. Alle drei Zusätze können als Bausteine für effektive Frontend-Lösungen für das Durchlaufen relationaler Daten verwendet werden.

Syntax:

```
Semantic( loadstatement | selectstatement)
```

Ein „semantic“-Ladevorgang erwartet eine Eingabe, die genau drei oder vier Felder umfasst. Dabei ist streng definiert, was jedes geordnete Feld darstellt, wie in der Tabelle unten gezeigt:

Felder für „semantic“-Ladevorgang

Feldname	Feldbeschreibung
1. Feld:	Dieses Tag ist eine Darstellung der ersten beiden Objekte, zwischen denen eine Beziehung besteht.
2. Feld:	Dieses Tag dient zur Beschreibung der „Vorwärts“-Beziehung zwischen dem ersten und dem zweiten Objekt. Wenn das erste Objekt untergeordnet und das zweite Objekt übergeordnet ist, können Sie einen Beziehungs-Tab erstellen, der „parent“ oder „parent of“ angibt, genau wie wenn Sie einer Beziehung von einem untergeordneten zu einem übergeordneten Element folgen.
3. Feld:	Dieses Tag ist eine Darstellung der nächsten beiden Objekte, zwischen denen eine Beziehung besteht.
4. Feld:	Dieses Feld ist optional. Dieses Tag dient zur Beschreibung der „Rückwärts“- bzw. „Umkehr“-Beziehung zwischen dem ersten und dem zweiten Objekt. Wenn das erste Objekt untergeordnet und das zweite Objekt übergeordnet ist, kann ein Beziehungs-Tab „child“ oder „child of“ angeben, genau wie wenn Sie einer Beziehung von einem übergeordneten zu einem untergeordneten Element folgen. Wenn Sie kein viertes Feld hinzufügen, wird das zweite Feld-Tag für die Beschreibung der Beziehung in beide Richtungen verwendet. In diesem Fall wird ein Pfeilsymbol automatisch als Teil des Tags hinzugefügt.

Der folgende Code ist ein Beispiel für den Zusatz `semantic`.

```
Semantic
Load
Object,
‘Parent’ AS Relationship,
NeighbouringObject AS Object,
‘Child’ AS Relationship
from graphdata.csv;
```

Es ist eine zulässige und typische Praxis, das dritte Feld genauso wie das erste Feld zu beschriften. Dadurch wird ein selbstreferenzierendes Lookup erstellt, sodass Sie Objekten zu den zugehörigen Objekten im jeweils nächsten Beziehungsschritt folgen können. Wenn das dritte Feld nicht den gleichen Namen hat, ist das Endergebnis ein einfacher Lookup von Objekten zu den direkten relationalen Nachbarn im Abstand von einem Schritt, eine wenig hilfreiche Ausgabe.

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Verwandte Funktionen

Funktionen	Interaktion
Hierarchy (page 66)	Der Ladezusatz „Hierarchy“ wird verwendet, um Knoten in übergeordneten und untergeordneten sowie anderen diagrammähnlichen Strukturen zu unterteilen und zu organisieren, und um sie in Tabellen umzuwandeln.
HierarchyBelongsTo (page 68)	Der Ladezusatz „HierarchyBelongsTo“ wird verwendet, um Vorfahren von übergeordneten und untergeordneten sowie anderen diagrammähnlichen Strukturen zu untersuchen und zu organisieren, und um sie in Tabellen umzuwandeln.

Beispiel – Erstellen eines besonderen Felds für Verbindungsbeziehungen mit dem Zusatz „semantic“

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz mit Einträgen für geografische Beziehungen, der in eine Tabelle namens `GeographyTree` geladen wird.

- Jeder Eintrag hat eine „ID“ am Beginn der Zeile und eine „ParentID“ am Ende der Zeile.

- Der Zusatz `semantic` fügt ein Feld mit besonderem Verhalten mit dem Namen `Relation` hinzu.

Ladeskript

```
GeographyTree:
LOAD
    ID,
    Geography,
    if(ParentID='',null(),ParentID) AS ParentID

INLINE [
ID,Geography,ParentID
1,World
2,Europe,1
3,Asia,1
4,North America,1
5,South America,1
6,UK,2
7,Germany,2
8,Sweden,2
9,South Korea,3
10,North Korea,3
11,China,3
12,London,6
13,Birmingham,6
];

SemanticTable:
Semantic Load
       ID as ID,
       'Parent' as Relation,
       ParentID as ID,
       'Child' as Relation
resident GeographyTree;
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `Id`
- `Geography`

Dann erstellen Sie ein Filterfenster mit `Relation` als Dimension. Klicken Sie auf **Bearbeitung fertig**.

Ergebnistabelle

Id	Geografie
1	Welt
2	Europa

Id	Geografie
3	Asien
4	Nordamerika
5	Südamerika
6	UK
7	Deutschland
8	Sweden
9	Südkorea
10	Nordkorea
11	China
12	London
13	Birmingham

Filterfenster

Beziehung
Untergeordnet
Übergeordnet

Klicken Sie auf **Europa** in der Dimension `Geography` in der Tabelle und klicken Sie auf **Untergeordnet** in der Dimension `Relation` im Filterfenster. Beachten Sie das erwartete Ergebnis in der Tabelle:

Ergebnistabelle mit untergeordneten Elementen für Europa

Id	Geografie
6	UK
7	Deutschland
8	Sweden

Durch erneutes Klicken auf **Untergeordnet** werden Orte angezeigt, die untergeordnete Elemente für UK sind und sich einen Schritt weiter unten befinden.

Ergebnistabelle mit untergeordneten Elementen für UK

Id	Geografie
12	London
13	Birmingham

Unless

Der Zusatz **unless** definiert eine Bedingung für die Evaluierung eines Befehls bzw. eines exit-Befehls. Es kann somit als Kurzform des vollständigen Befehls **if..end if** betrachtet werden.

Syntax:

```
(Unless condition statement | exitstatement Unless condition )
```

statement oder **exitstatement** wird nur ausgeführt, wenn **condition** False ergibt.

Der Zusatz **unless** kann auch für Befehle verwendet werden, die bereits einen oder mehrere andere Befehle aufweisen, einschließlich zusätzlicher **when**- oder **unless**-Zusätze.

Argumente

Argument	Beschreibung
condition	Eine logische Formel, die True oder False ergibt.
statement	Ein beliebiger Qlik Sense-Skriptbefehl mit Ausnahme von Steuerungsbefehlen.
exitstatement	Eine **exit for**-, **exit do**- oder **exit sub**-Bedingung oder ein **exit script**-Befehl.

Verwendung

Der Befehl `Unless` gibt einen booleschen Wert zurück. In der Regel wird diese Art Funktion als Bedingung verwendet, wenn der Benutzer Teile des Skripts bedingt laden oder ausschließen möchte.

Die folgenden Zeilen zeigen drei Beispiele, wie die Funktion `Unless` verwendet werden kann:

```
exit script unless A=1;

unless A=1 LOAD * from myfile.csv;

unless A=1 when B=2 drop table Tab1;
```

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Beispiel 1 – Präfix „unless“

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Erstellung einer Variable A, die einen Wert von 1 erhält
- Datensatz, der in eine Tabelle namens „Transactions“ geladen wird, es sei denn die Variable A = 2

Ladeskript

```
LET A = 1;

UNLESS A = 2

Transactions:
LOAD
*
Inline [
id, date, amount
1, 08/30/2018, 23.56
2, 09/07/2018, 556.31
3, 09/16/2018, 5.75
4, 09/22/2018, 125.00
5, 09/22/2018, 484.21
6, 09/22/2018, 59.18
7, 09/23/2018, 177.42
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `id`
- `date`
- `amount`

Ergebnistabelle

id	date	amount
1	08/30/2018	23.56
2	09/07/2018	556.31
3	09/16/2018	5.75

id	date	amount
4	09/22/2018	125.00
5	09/22/2018	484.21
6	09/22/2018	59.18
7	09/23/2018	177.42

Da der Variablen A am Beginn des Skripts der Wert von 1 zugewiesen ist, wird die Bedingung nach dem Präfix `Unless` ausgewertet und gibt ein Ergebnis von `FALSE` zurück. Infolgedessen führt das Skript den Befehl `Load` weiter aus. In der Ergebnistabelle werden alle Einträge aus der Tabelle `Transactions` angezeigt.

Wenn dieser Variablenwert auf 2 festgelegt wird, werden keine Daten in das Datenmodell geladen.

Beispiel 2 – Suffix „unless“

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript startet damit, dass ein anfänglicher Datensatz in eine Tabelle namens `Transactions` geladen wird. Das Skript wird dann beendet, es sei denn, es sind weniger als 10 Einträge in der Tabelle `Transactions` vorhanden.

Wenn diese Bedingung nicht zu einer Beendigung des Skripts führt, wird eine weitere Reihe von Transaktionen in der Tabelle `Transactions` zusammengeführt, und dieser Prozess wird wiederholt.

Ladeskript

```
Transactions:
LOAD
*
Inline [
id, date, amount
1, 08/30/2018, 23.56
2, 09/07/2018, 556.31
3, 09/16/2018, 5.75
4, 09/22/2018, 125.00
5, 09/22/2018, 484.21
6, 09/22/2018, 59.18
7, 09/23/2018, 177.42
];


exit script unless NoOfRows('Transactions') < 10 ;

Concatenate
LOAD
*
```

```
Inline [
id, date, amount
8, 10/01/2018, 164.27
9, 10/03/2018, 384.00
10, 10/06/2018, 25.82
11, 10/09/2018, 312.00
12, 10/15/2018, 4.56
13, 10/16/2018, 90.24
14, 10/18/2018, 19.32
];

exit script unless NoOfRows('Transactions') < 10 ;

Concatenate
LOAD
*
Inline [
id, date, amount
15, 10/01/2018, 164.27
16, 10/03/2018, 384.00
17, 10/06/2018, 25.82
18, 10/09/2018, 312.00
19, 10/15/2018, 4.56
20, 10/16/2018, 90.24
21, 10/18/2018, 19.32
];

exit script unless NoOfRows('Transactions') < 10 ;
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- id
- date
- amount

Ergebnistabelle

id	date	amount
1	08/30/2018	23.56
2	09/07/2018	556.31
3	09/16/2018	5.75
4	09/22/2018	125.00
5	09/22/2018	484.21
6	09/22/2018	59.18

id	date	amount
7	09/23/2018	177.42
8	10/01/2018	164.27
9	10/03/2018	384.00
10	10/06/2018	25.82
11	10/09/2018	312.00
12	10/15/2018	4.56
13	10/16/2018	90.24
14	10/18/2018	19.32

Es gibt sieben Einträge in jedem der drei Datensätze des Ladeskripts.

Der erste Datensatz (mit Transaktions-`id` 1 bis 7) wird in die Anwendung geladen. Die Bedingung `Unless` wertet aus, ob weniger als 10 Zeilen in der Tabelle `Transactions` vorhanden sind. Sie wird zu `TRUE` ausgewertet, und daher wird der zweite Datensatz (mit Transaktions-`id` 8 bis 14) in die Anwendung geladen. Die zweite Bedingung `Unless` wertet aus, ob weniger als 10 Einträge in der Tabelle `Transactions` vorhanden sind. Die Auswertung ergibt `FALSE`, und damit endet das Skript.

Beispiel 3 – Mehrere Präfixe „unless“

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

In diesem Beispiel wird ein Datensatz, der eine Transaktion enthält, als Tabelle namens `Transactions` erstellt. Dann wird eine „for“-Schleife ausgelöst, in der zwei verschachtelte „unless“-Befehle Folgendes auswerten:

1. Es sei denn, in der Tabelle `Transactions` sind mehr als 100 Einträge vorhanden
2. Es sei denn, die Anzahl der Einträge in der Tabelle `Transactions` ist ein Vielfaches von 6

Wenn diese Bedingungen `FALSE` sind, werden weitere sieben Einträge generiert und in der vorhandenen Tabelle `Transactions` zusammengeführt. Dieser Vorgang wird wiederholt, bis eine der zwei Transaktionen einen Wert von `TRUE` zurückgibt.

Ladeskript

```
Transactions:
Load
        0 as id
Autogenerate 1;

For i = 1 to 100
        unless NoOfRows('Transactions') > 100 unless mod(NoOfRows('Transactions'),6) = 0
```

```
                Concatenate
      Load
if(isnull(Peek(id)),1,peek(id)+1) as id
                Autogenerate 7;
next i
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu:`id`.

Ergebnistabelle

id
0
1
2
3
4
5
+30 weitere Zeilen

Die verschachtelten „unless“-Befehle, die in der „for“-Schleife vorhanden sind, werden wie folgt ausgewertet:

1. Sind mehr als 100 Zeilen in der Tabelle `Transactions` vorhanden?
2. Ist die Anzahl der Einträge in der Tabelle `Transactions` ein Vielfaches von 6?

Immer, wenn beide „unless“-Befehle einen Wert von FALSE zurückgeben, werden weitere sieben Einträge generiert und in der vorhandenen Tabelle `Transactions` zusammengeführt.

Diese Befehle geben fünf Mal einen Wert von FALSE zurück. An dieser Stelle befinden sich insgesamt 36 Datenzeilen in der Tabelle `Transactions`.

Danach gib der zweite `Unless`-Befehl einen Wert von TRUE zurück, und daher wird die load-Anweisung danach nicht mehr ausgeführt.

When

Der Zusatz **when** definiert eine Bedingung für die Ausführung eines Befehls bzw. eines exit-Befehls. Es kann somit als Kurzform des vollständigen Befehls **if..end if** betrachtet werden.

Syntax:

```
(when condition statement | exitstatement when condition )
```

Rückgabe Datentyp: Boolesch

In Qlik Sense wird der boolesche Wert „wahr“ durch -1 dargestellt, der Wert „falsch“ durch 0.

statement oder **exitstatement** wird nur ausgeführt, wenn die Bedingung TRUE ergibt.

Der Zusatz `when` kann auch für Befehle verwendet werden, die bereits einen oder mehrere andere Befehle aufweisen, einschließlich zusätzlicher `when`- oder `Unless`-Zusätze.

Verwendung

Der Befehl `when` gibt einen booleschen Wert zurück. In der Regel wird diese Art Funktion als Bedingung verwendet, wenn der Benutzer Teile des Skripts laden oder ausschließen möchte.

Argumente

Argument	Beschreibung
condition	Eine logische Formel, die TRUE oder FALSE ergibt.
statement	Ein beliebiger Qlik Sense-Skriptbefehl mit Ausnahme von Steuerungsbefehlen.
exitstatement	Eine **exit for**-, **exit do**- oder **exit sub**-Bedingung oder ein **exit script**-Befehl.

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Funktionsbeispiele

Beispiel	Ergebnis
`exit script when A=1;`	Wenn der Befehl `A=1` als TRUE ausgewertet wird, hält das Skript an.
`when A=1 LOAD * from myfile.csv;`	Wenn der Befehl `A=1` als TRUE ausgewertet wird, wird die Datei `myfile.csv` geladen.
`when A=1 unless B=2 drop table Tab1;`	Wenn der Befehl `A=1` als TRUE und `B=2` als FALSE ausgewertet wird, wird die Tabelle `Tab1` verworfen.

Beispiel 1 – Präfix „when“

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz mit Datumswerten und Beträgen wird an eine Tabelle namens „`Transactions`“ gesendet.
- Der Befehl `Let` gibt an, dass die Variable `A` erstellt wird und den Wert `1` hat.
- Die Bedingung `When` gibt an, dass das Skript weiter geladen wird, wenn `A` dem Wert `1` entspricht.

Ladeskript

```
LET A = 1;

WHEN A = 1

Transactions:
LOAD
*
Inline [
id, date, amount
1, 08/30/2018, 23.56
2, 09/07/2018, 556.31
3, 09/16/2018, 5.75
4, 09/22/2018, 125.00
5, 09/22/2018, 484.21
6, 09/22/2018, 59.18
7, 09/23/2018, 177.42
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `id`
- `date`
- `amount`

Ergebnistabelle

id	date	amount
1	08/30/2018	23.56
2	09/07/2018	556.31

id	date	amount
3	09/16/2018	5.75
4	09/22/2018	125.00
5	09/22/2018	484.21
6	09/22/2018	59.18
7	09/23/2018	177.42

Da der Variablen A am Beginn des Skripts der Wert von 1 zugewiesen ist, wird die Bedingung nach dem Präfix when ausgewertet und gibt ein Ergebnis von TRUE zurück. Da ein Ergebnis von TRUE zurückgegeben wird, führt das Skript die load-Anweisung weiter aus. In der Ergebnistabelle werden alle Einträge angezeigt.

Wenn dieser Variablenwert auf einen beliebigen Wert ungleich 1 festgelegt wird, werden keine Daten in das Datenmodell geladen.

Beispiel 2 – Suffix „When“

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Drei Datensätze mit Datumswerten und Beträgen werden an eine Tabelle namens „Transactions“ gesendet.
    - Der erste Datensatz enthält die Transaktionen 1-7.
    - Der zweite Datensatz enthält die Transaktionen 8-14.
    - Der dritte Datensatz enthält die Transaktionen 15-21.
- Eine when-Bedingung entscheidet, ob die Tabelle „Transactions“ mehr als zehn Zeilen enthält. Wenn einer der Befehle when als TRUE ausgewertet wird, hält das Ladeskript an. Diese Bedingung wird am Ende jedes der drei Datensätze platziert.

Ladeskript

```
Transactions:
LOAD
*
Inline [
id, date, amount
1, 08/30/2018, 23.56
2, 09/07/2018, 556.31
3, 09/16/2018, 5.75
4, 09/22/2018, 125.00
5, 09/22/2018, 484.21
6, 09/22/2018, 59.18
7, 09/23/2018, 177.42
```

```
];

exit script when NoOfRows('Transactions') > 10 ;

Concatenate
LOAD
*
Inline [
id, date, amount
8, 10/01/2018, 164.27
9, 10/03/2018, 384.00
10, 10/06/2018, 25.82
11, 10/09/2018, 312.00
12, 10/15/2018, 4.56
13, 10/16/2018, 90.24
14, 10/18/2018, 19.32
];

exit script when NoOfRows('Transactions') > 10 ;

Concatenate
LOAD
*
Inline [
id, date, amount
15, 10/01/2018, 164.27
16, 10/03/2018, 384.00
17, 10/06/2018, 25.82
18, 10/09/2018, 312.00
19, 10/15/2018, 4.56
20, 10/16/2018, 90.24
21, 10/18/2018, 19.32
];

exit script when NoOfRows('Transactions') > 10 ;
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `id`
- `date`
- `amount`

Ergebnistabelle

id	date	amount
1	08/30/2018	23.56
2	09/07/2018	556.31

id	date	amount
3	09/16/2018	5.75
4	09/22/2018	125.00
5	09/22/2018	484.21
6	09/22/2018	59.18
7	09/23/2018	177.42
8	10/01/2018	164.27
9	10/03/2018	384.00
10	10/06/2018	25.82
11	10/09/2018	312.00
12	10/15/2018	4.56
13	10/16/2018	90.24
14	10/18/2018	19.32

In jedem der drei Datensätze befinden sich sieben Transaktionen. Der erste Datensatz mit den Transaktionen 1 bis 7 wird in die Anwendung geladen. Die `When`-Bedingung nach dieser load-Anweisung wird als FALSE ausgewertet, weil die Tabelle „`Transactions`“ weniger als zehn Zeilen enthält. Das Ladeskript fährt mit dem nächsten Datensatz fort.

Der zweite Datensatz mit den Transaktionen 8-14 wird in die Anwendung geladen. Die zweite `When`-Bedingung wird als TRUE ausgewertet, weil mehr als zehn Zeilen in der Tabelle „`Transactions`“ enthalten sind. Daher wird das Skript beendet.

Beispiel 3 – Mehrere Präfixe „When“

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz, der eine einzelne Transaktion enthält, wird als Tabelle namens „`Transactions`“ erstellt.
- Eine `For`-Schleife wird ausgelöst, die zwei verschachtelte `When`-Bedingungen enthält. Diese werten aus, ob:
    1. weniger als 100 Einträge in der Tabelle „`Transactions`“ vorhanden sind.
    2. die Anzahl der Einträge in der Tabelle „`Transactions`“ kein Vielfaches von 6 sind.

Ladeskript

```
RowsCheck = NoOfRows('Transactions') < 100 or mod(NoOfRows('Transactions'),6) <> 0;
Transactions:
```

```
Load
       0 as id
Autogenerate 1;
For i = 1 to 100
       when(RowsCheck)
              Concatenate
       Load
              if(isnull(Peek(id)),1,peek(id)+1) as id
       Autogenerate 7;
next i
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu:

- `id`

Die Ergebnistabelle zeigt nur die ersten fünf Transaktions-IDs, aber das Ladeskript erstellt 36 Zeilen und endet, nachdem die `When`-Bedingung erfüllt ist.

Ergebnistabelle

id
0
1
2
3
4
5
+30 weitere Zeilen

Die verschachtelten `When`-Bedingungen in der `For`-Schleife werten die folgenden Fragen aus:

- Sind weniger als 100 Zeilen in der Tabelle „`Transactions`“ vorhanden?
- Ist die Gesamtzahl der Einträge in der Tabelle „`Transactions`“ ein Vielfaches von 6?

Immer, wenn beide `When`-Bedingungen einen Wert von TRUE zurückgeben, werden weitere sieben Einträge generiert und in der vorhandenen Tabelle „`Transactions`“ zusammengeführt.

Die `When`-Bedingungen geben fünf Mal einen Wert von TRUE zurück. An dieser Stelle befinden sich insgesamt 36 Datenzeilen in der Tabelle „`Transactions`“.

Nachdem 36 Datenzeilen in der Tabelle „`Transactions`“ erstellt wurden, gibt der zweite `When`-Befehl einen Wert von FALSE zurück, und daher wird die load-Anweisung danach nicht mehr ausgeführt.

3.3 Reguläre Skriptbefehle

Gewöhnliche Befehle dienen dazu, Daten einzulesen und diese Daten zu strukturieren oder zu verändern. Sie können sich über mehrere Zeilen erstrecken und müssen stets mit einem Semikolon enden , ";".

Sämtliche Skriptbefehle können in Groß- oder Kleinbuchstaben oder einer Kombination aus beiden eingegeben werden. Bei Feld- und Variablennamen, die im Skript vorkommen, wird Groß- und Kleinschreibung jedoch unterschieden.

Reguläre Skriptbefehle – Übersicht

Jede Funktion wird nach der Übersicht genauer beschrieben. Sie können auch auf den Funktionsnamen in der Syntax klicken, um direkt auf die Details zu der spezifischen Funktion zuzugreifen.

Alias
Die **alias**-Anweisung definiert einen Aliasnamen für ein Feld. Wann immer dieses Feld im Skript vorkommt, wird es umbenannt.

```
Alias fieldname as aliasname {,fieldname as aliasname}
```

Autonumber
Diese Anweisung erstellt eine eindeutige ganze Zahl für jeden distinkten ausgewerteten Wert in einem Feld, der bei der Skriptausführung gefunden wird.

```
AutoNumber fields [Using namespace] ]
```

Binary
Mit dem Befehl **binary** werden die Daten aus einem anderen QlikView-Dokument geladen, einschließlich der Section Access-Daten.

```
Binary  [path] filename
```

comment
Durch diesen Befehl können Sie Kommentare zu Feldern (Metadaten) selbst eingeben oder aus Datenbanken und Tabellen einlesen. Feldnamen, die in der App nicht vorkommen, werden dabei ignoriert. Wenn ein Feldname mehrmals vorkommt, wird der letzte Wert verwendet.

```
Comment field *fieldlist using mapname
Comment field fieldname with comment
```

comment table
Durch diesen Befehl können Sie Kommentare zu Tabellen (Metadaten) selbst eingeben oder aus Datenbanken und Tabellen einlesen.

```
Comment table tablelist using mapname
Comment table tablename with comment
```

Connect

Diese Funktion ist in Qlik Sense SaaS nicht verfügbar.

Der Befehl **CONNECT** legt den Qlik Sense-Zugriff auf eine allgemeine Datenbank über die OLE DB/ODBC-Schnittstelle fest. Für ODBC muss die Datenquelle zunächst mithilfe des ODBC-Administrators angegeben werden.

```
ODBC Connect TO connect-string [ ( access_info ) ]
OLEDB CONNECT TO connect-string [ ( access_info ) ]
CUSTOM CONNECT TO connect-string [ ( access_info ) ]
LIB CONNECT TO connection
```

Declare

Der Befehl **Declare** wird verwendet, um Felddefinitionen zu erstellen, in denen Sie Beziehungen zwischen Feldern und Funktionen festlegen können. Mithilfe mehrerer Felddefinitionen lassen sich automatisch abgeleitete Felder erstellen, die als Dimensionen verwendet werden können. Sie können beispielsweise eine Kalenderdefinition erstellen und diese zur Generierung von damit in Bezug stehenden Dimensionen, wie Jahr, Monat, Woche und Tag, aus einem Datumsfeld verwenden.

```
definition_name:
Declare [Field[s]] Definition [Tagged tag_list ]
[Parameters parameter_list ]
Fields field_list
[Groups group_list ]

<definition name>:
Declare [Field][s] Definition
Using <existing_definition>
[With <parameter_assignment> ]
```

Derive

Der Befehl **Derive** wird zur Generierung abgeleiteter Felder auf Grundlage einer Felddefinition verwendet, die mit einem **Declare**-Befehl erstellt wurde. Sie können entweder angeben, für welche Datenfelder die Felder abgeleitet werden sollen, oder die Felder explizit oder implizit auf Grundlage der Tags für die Felder ableiten.

```
Derive [Field[s]] From [Field[s]] field_list Using definition
Derive [Field[s]] From Explicit [Tag[s]] (tag_list) Using definition
Derive [Field[s]] From Implicit [Tag[s]] Using definition
```

Direct Query

Mit dem Befehl **DIRECT QUERY** können Sie Tabellen über eine ODBC- oder OLE DB-Verbindung aufrufen, und zwar mithilfe der Funktion Direct Discovery.

```
Direct Query [path]
```

Directory

Die **Directory**-Anweisung definiert, welches Verzeichnis in den nachfolgenden **LOAD**-Anweisungen nach Datendateien durchsucht wird, bis eine neue **Directory**-Anweisung erstellt wird.

```
Directory [path]
```

Disconnect

Der **Disconnect**-Befehl beendet die aktuelle ODBC/OLE DB/benutzerdefinierte Verbindung. Der Befehl ist optional.

```
Disconnect
```

drop field

Durch den Befehl **drop field** können jederzeit ein oder mehrere Qlik Sense-Felder während der Skriptausführung aus dem Datenmodell und damit aus dem Arbeitsspeicher gelöscht werden. Die Eigenschaft „distinct“ einer Tabelle wird nach einem **drop field**-Befehl entfernt.

*Sowohl **drop field** als auch **drop fields** ist zulässig. Die Funktion der beiden Befehle ist identisch. Ist kein Tabellenname angegeben, wird das Feld aus allen Tabellen gelöscht, in denen es vorkommt.*

```
Drop field  fieldname [ , fieldname2 ...] [from tablename1 [ , tablename2
...]]
drop fields fieldname [ , fieldname2 ...] [from tablename1 [ , tablename2
...]]
```

drop table

Durch den Befehl **drop table** können jederzeit eine oder mehrere interne Qlik Sense-Tabellen während der Skriptausführung aus dem Datenmodell und damit aus dem Arbeitsspeicher gelöscht werden.

*Es werden die Formulare **drop table** und **drop tables** akzeptiert.*

```
Drop table  tablename [, tablename2 ...]
drop tables[ tablename [, tablename2 ...]
```

Execute

Der Befehl **Execute** wird zur Ausführung anderer Programme verwendet, während Qlik Sense Daten lädt. Dies dient z. B. dazu, notwendige Konvertierungen vorzunehmen.

```
Execute commandline
```

FlushLog

Der Befehl **FlushLog** erzwingt, dass Qlik Sense den Inhalt des Skriptpuffers in die Skriptprotokolldatei schreibt.

```
FlushLog
```

Force

Der Befehl **force** erzwingt, dass Qlik Sense die Feldnamen und Feldwerte der folgenden **LOAD**- und **SELECT**-Befehle als in Großbuchstaben, in Kleinbuchstaben oder wie angegeben (gemischt) interpretiert. Auf diese Weise können Tabellen über Schlüsselfelder automatisch miteinander verknüpft werden, auch wenn die Feldnamen in den Datenquellen hinsichtlich Groß- und Kleinschreibung nicht übereinstimmen.

```
Force ( capitalization | case upper | case lower | case mixed )
```

LOAD

Der **LOAD**-Befehl lädt Felder aus einer Datei aus Daten, die im Skript definiert sind, aus einer zuvor geladenen Tabelle, aus einer Webseite, aus dem Ergebnis eines nachfolgenden **SELECT**-Befehls oder durch automatisches Generieren der Daten. Daten können auch aus Analyseverbindungen geladen werden.

```
 Load [ distinct ] *fieldlist
[( from file [ format-spec ] |
from_field fieldassource [format-spec]
inline data [ format-spec ] |
resident table-label |
autogenerate size )]
[ where criterion | while criterion ]
[ group_by groupbyfieldlist ]
[order_by orderbyfieldlist ]
[extension pluginname.functionname(tabledescription)]
```

Let

Der **let**-Befehl ergänzt den **set**-Befehl und definiert die Skriptvariablen. Im Gegensatz zum **set**-Befehl wird beim **let**-Befehl der Ausdruck rechts des Gleichheitszeichens "=" zur Laufzeit des Skripts ausgewertet, bevor er der Variablen zugewiesen wird.

```
Let variablename=expression
```

Loosen Table

Durch den Befehl **Loosen Table** können eine oder mehrere interne Qlik Sense-Datentabellen während der Skriptausführung explizit als freie Tabellen eingestuft werden. Bei einer freien Tabelle werden alle Verknüpfungen zwischen Feldwerten in der Tabelle entfernt. Ein ähnlicher Effekt kann durch Laden jedes Felds der freien Tabellen als unabhängige, unverbundene Tabellen erzielt werden. Freie Tabellen können während des Testens nützlich sein, um verschiedene Teile der Datenstruktur vorübergehend zu isolieren. Im Tabellenmodell werden freie Tabellen durch gepunktete Linien dargestellt. Enthält das Skript eine oder mehrere **Loosen Table**-Befehle, werden zu Beginn der Skriptausführung alle vorher getroffenen Einstellungen bezüglich freier Tabellen in Qlik Sense aufgehoben.

```
tablename [ , tablename2 ...]
Loosen Tables tablename [ , tablename2 ...]
```

Map ... using

Die Funktion **map ... using** wird für die Zuordnung eines bestimmten Feldwerts oder einer Formel zu den Werten einer bestimmten Mapping-Tabelle verwendet. Die Mapping-Tabelle wird mit dem Befehl **Mapping** erstellt.

```
Map  *fieldlist Using  mapname
```

NullAsNull

Der Befehl **NullAsNull** schaltet die Konvertierung von NULL-Werten in Stringwerte aus, die zuvor mit dem Befehl **NullAsValue** festgelegt wurde.

```
NullAsNull *fieldlist
```

NullAsValue

Mit dem Befehl **NullAsValue** werden die Felder angegeben, für die NULL-Werte in Werte konvertiert werden sollen.

```
NullAsValue *fieldlist
```

Qualify

Mit dem Befehl **Qualify** wird die Qualifikation von Feldnamen aktiviert, d. h. Feldnamen erhalten den Tabellennamen als Zusatz.

```
Qualify *fieldlist
```

Rem

Der **rem**-Befehl dient dazu, Anmerkungen oder Kommentare in das Skript einzufügen oder Befehle vorübergehend zu deaktivieren, ohne sie aus dem Skript zu löschen.

```
Rem string
```

Rename Field

Diese Skriptfunktion benennt nach dem Laden ein oder mehrere bestehende Qlik Sense-Felder um.

```
Rename field (using mapname|oldname to newname{ , oldname to newname })
```

```
Rename Fields (using mapname | oldname to newname{ , oldname to newname })
```

Rename Table

Diese Skriptfunktion benennt nach dem Laden eine oder mehrere bestehende interne Qlik Sense-Tabellen um.

```
Rename table  (using mapname | oldname to newname{ , oldname to newname })
Rename Tables (using mapname | oldname to newname{ , oldname to newname })
```

Section

Mit dem **section**-Befehl können Sie definieren, ob die nachfolgenden Befehle **LOAD** und **SELECT** Daten laden oder aber eine Zugriffskontrollen festlegen.

```
Section (access | application)
```

Select

Die Auswahl von Feldern aus einer ODBC-Datenquelle oder aus einem OLE DB-Provider erfolgt über die gewöhnlichen SQL-**SELECT**-Befehle. Allerdings hängt die Akzeptanz von **SELECT**-Befehlen vom verwendeten ODBC-Treiber oder OLE DB-Provider ab.

```
Select [all | distinct | distinctrow | top n [percent] ] *fieldlist
```

```
From tablelist

[Where criterion ]

[Group by fieldlist [having criterion ] ]

[Order by fieldlist [asc | desc] ]

[ (Inner | Left | Right | Full)Join tablename on fieldref = fieldref ]
```

Set
Der Befehl **set** wird zum Festlegen der Skriptvariablen verwendet. Diese können Strings, Pfade, Laufwerke usw. im Skript ersetzen.

```
Set variablename=string
```

Sleep
Der Befehl **sleep** hält die Ausführung des Skripts für eine festgelegte Zeit an.

```
Sleep n
```

SQL
Mithilfe der **SQL**-Anweisung können Sie einen beliebigen SQL-Befehl über eine ODBC- oder OLE DB-Verbindung senden.

```
SQL sql_command
```

SQLColumns
Der Befehl **sqlcolumns** erzeugt eine Reihe von Feldern, welche die Spalten der ODBC- oder OLE DB-Datenquelle beschreiben, zu der mit dem **connect**-Befehl eine Verbindung hergestellt wurde.

```
SQLColumns
```

SQLTables
Der Befehl **sqltables** erzeugt eine Reihe von Feldern, welche die Tabellen einer ODBC- oder OLE DB-Datenquelle beschreiben, zu der mit dem **connect**-Befehl eine Verbindung hergestellt wurde.

```
SQLTables
```

SQLTypes
Der Befehl **sqltypes** erzeugt eine Reihe von Feldern, welche die Arten der ODBC- oder OLE DB-Datenquelle beschreiben, zu der mit dem **connect**-Befehl eine Verbindung hergestellt wurde.

```
SQLTypes
```

Star
Der String, der alle Werte eines Feldes repräsentieren soll (Stern-Symbol), kann mit dem **star**-Befehl definiert werden. Dies hat Gültigkeit für alle nachfolgenden **LOAD**- und **SELECT**-Befehle.

```
Star is [ string ]
```

Store
Mit dem Befehl **Store** wird eine QVD-, Parquet-, CSV- oder TXT-Datei erstellt.

```
Store [ *fieldlist from] table into filename [ format-spec ];
```

Tag

Dieser Skriptbefehl bietet eine Möglichkeit zum Zuweisen von Tags zu einem oder mehreren Feldern oder Tabellen. Wird der Versuch unternommen, einem Feld oder einer Tabelle ein Tag hinzuzufügen, das bzw. die nicht in der App vorhanden ist, wird dieser Vorgang ignoriert. Gibt es Konflikte durch mehrfach vorkommende Feldnamen oder Tags, wird der letzte Tag verwendet.

```
Tag[field|fields] fieldlist with tagname
Tag [field|fields] fieldlist using mapname
Tag table tablelist with tagname
```

Trace

Mit dem Befehl **trace** kann ein String im Fenster **Status der Skriptausführung** angezeigt und in die log-Datei geschrieben werden. Er eignet sich besonders zum Debugging. Wenn Sie die $-Erweiterungen von Variablen verwenden, die vor dem **trace**-Befehl berechnet werden, können Sie die Meldung anpassen.

```
Trace string
```

Unmap

Mit dem Befehl **Unmap** wird der Feldwert Mapping deaktiviert, der mit dem Befehl **Map ... Using** für die anschließend geladenen Felder angegeben wurde.

```
Unmap *fieldlist
```

Unqualify

Die **Unqualify**-Anweisung wird verwendet, um die Qualifizierung von Feldnamen, die zuvor durch die **Qualify**-Anweisung aktiviert wurden, zu deaktivieren.

```
Unqualify *fieldlist
```

Untag

Dieser Skriptbefehl bietet eine Möglichkeit zum Entfernen von Tags aus Feldern oder Tabellen. Wird der Versuch unternommen, ein Tag aus einem Feld oder einer Tabelle zu entfernen, das bzw. die nicht in der App vorhanden ist, wird dieser Vorgang ignoriert.

```
Untag[field|fields] fieldlist with tagname
Tag [field|fields] fieldlist using mapname
Tag table tablelist with tagname
```

Alias

Die **alias**-Anweisung definiert einen Aliasnamen für ein Feld. Wann immer dieses Feld im Skript vorkommt, wird es umbenannt.

Syntax:

```
alias fieldname as aliasname {,fieldname as aliasname}
```

Argumente:

Argumente

Argument	Beschreibung
fieldname	Der Name des Felds in der Datenquelle
aliasname	Ein Aliasname, den Sie stattdessen verwenden möchten

Beispiele und Ergebnisse:

Beispiel	Ergebnis
`Alias ID_N as NameID;`	
`Alias A as Name, B as Number, C as Date;`	Die Umbenennung erfolgt in allen nachfolgenden **SELECT**- und **LOAD**-Befehlen. Mit einem weiteren **alias**-Befehl kann ein neuer Aliasname für einen Feldnamen an einer beliebigen nachfolgenden Stelle im Skript definiert werden.

AutoNumber

Diese Anweisung erstellt eine eindeutige ganze Zahl für jeden distinkten ausgewerteten Wert in einem Feld, der bei der Skriptausführung gefunden wird.

Sie können auch die *autonumber (page 597)*-Funktion innerhalb einer **LOAD**-Anweisung verwenden; dies weist aber bestimmte Einschränkungen auf, wenn Sie eine optimierte Ladung verwenden möchten. Sie können eine optimierte Ladung erstellen, indem Sie die Daten zuerst aus einer **QVD**-Datei laden und dann die **AutoNumber**-Anweisung verwenden, um die Werte zu Symbolschlüsseln zu konvertieren.

Syntax:

```
AutoNumber *fieldlist [Using namespace] ]
```

Argumente:

Argumente

Argument	Beschreibung
*fieldlist	Eine kommagetrennte Liste der Felder, deren Werte durch einen eindeutigen Ganzzahlwert ersetzt werden sollen. Sie können die Platzhalterzeichen ? und * in den Feldnamen verwenden, um alle Felder mit übereinstimmenden Namen einzuschließen. Sie können auch * verwenden, um alle Felder einzuschließen. Wenn Platzhalter verwendet werden, müssen Sie die Feldnamen in Anführungszeichen einschließen.

Argument	Beschreibung
namespace	**Using** namespace ist optional. Sie können diese Option verwenden, wenn Sie ein Namespace erstellen möchten, in dem identische Werte in verschiedenen Feldern den gleichen Schlüssel verwenden. Wenn Sie diese Option nicht verwenden, haben alle Felder einen getrennten Schlüsselindex.

Beschränkungen:

Wenn Sie mehrere **LOAD**-Anweisungen im Skript haben, müssen Sie die **AutoNumber**-Anweisung nach der letzten **LOAD**-Anweisung platzieren.

Beispiel: Skript mit AutoNumber

Skriptbeispiel

In diesem Beispiel werden die Daten zuerst ohne den Befehl **AutoNumber** geladen. Dann wird der Befehl **AutoNumber** hinzugefügt, um die Auswirkung zu zeigen.

Im Beispiel verwendete Daten

Laden Sie die folgenden Daten als Inline-Ladevorgang in den Dateneditor, um das folgende Skriptbeispiel zu erstellen. Lassen Sie den Befehl **AutoNumber** vorläufig auskommentiert.

```
RegionSales:
LOAD *,
Region &'|'& Year &'|'& Month as KeyToOtherTable
INLINE
[ Region, Year, Month, Sales
North, 2014,   May,    245
North, 2014,   May,    347
North, 2014,   June,   127
South, 2014,   June,   645
South, 2013,   May,    367
South, 2013,   May,    221
];

Budget:
LOAD Budget,
Region &'|'& Year &'|'& Month as KeyToOtherTable
INLINE
[Region, Year, Month, Budget
North,  2014,   May,    200
North,  2014,   May,    350
North,  2014,   June,   150
South,  2014,   June,   500
South,  2013,   May,    300
South,  2013,   May,    200
];
```

```
//AutoNumber KeyToOtherTable;
```

Visualisierungen erstellen

Erstellen Sie zwei Tabellenvisualisierungen in einem Qlik Sense-Arbeitsblatt. Fügen Sie **KeyToOtherTable**, **Region**, **Year**, **Month** und **Sales** als Dimensionen zur ersten Tabelle hinzu. Fügen Sie **KeyToOtherTable**, **Region**, **Year**, **Month** und **Budget** als Dimensionen zur zweiten Tabelle hinzu.

Ergebnis

Tabelle „RegionSales“

KeyToOtherTable	Region	Year	Month	Sales
North|2014|June	North	2014	June	127
North|2014|May	North	2014	May	245
North|2014|May	North	2014	May	347
South|2013|May	South	2013	May	221
South|2013|May	South	2013	May	367
South|2014|June	South	2014	June	645

Tabelle „Budget“

KeyToOtherTable	Region	Year	Month	Budget
North|2014|June	North	2014	June	150
North|2014|May	North	2014	May	200
North|2014|May	North	2014	May	350
South|2013|May	South	2013	May	200
South|2013|May	South	2013	May	300
South|2014|June	South	2014	June	500

Erläuterung

Das Beispiel zeigt ein zusammengesetztes Feld **KeyToOtherTable**, das die beiden Tabellen verknüpft. **AutoNumber** wird nicht verwendet. Beachten Sie die Länge der Werte für **KeyToOtherTable**.

Befehl „AutoNumber“ hinzufügen

Kommentieren Sie den Befehl **AutoNumber** im Ladeskript aus.

```
AutoNumber KeyToOtherTable;
```

Ergebnis

Tabelle „RegionSales“

KeyToOtherTable	Region	Year	Month	Sales
1	North	2014	June	127
1	North	2014	May	245
2	North	2014	May	347
3	South	2013	May	221
4	South	2013	May	367
4	South	2014	June	645

Tabelle „Budget“

KeyToOtherTable	Region	Year	Month	Budget
1	North	2014	June	150
1	North	2014	May	200
2	North	2014	May	350
3	South	2013	May	200
4	South	2013	May	300
4	South	2014	June	500

Erläuterung

Die Feldwerte für **KeyToOtherTable** wurden durch eindeutige Ganzzahlwerte ersetzt. Dadurch wurde die Länge der Feldwerte reduziert und Arbeitsspeicher gespart. Die Schlüsselfelder in beiden Feldern sind von **AutoNumber** betroffen, und die Tabellen bleiben verknüpft. Das Beispiel ist für Demonstrationszwecke nur kurz, wäre aber besonders in einer Tabelle mit vielen Zeilen sinnvoll.

Binary

Mit dem Befehl **binary** werden die Daten aus einer anderen Qlik Sense-App oder einem Dokument von QlikView einschließlich der Section Access-Daten geladen. Andere Elemente der App sind nicht inbegriffen, z. B. Arbeitsblätter, Stories, Visualisierungen, Master-Elemente oder Variablen.

In einem Skript darf jeweils nur ein **binary**-Befehl vorkommen. Der **binary**-Befehl muss ganz am Anfang stehen, sogar vor den SET-Befehlen, die sich normalerweise am Anfang des Skripts befinden.

Syntax:

```
binary [path] filename
```

Argumente:

Argumente

Argument	Beschreibung
path	Der Pfad zur Datei, der ein Verweis zu einer Ordner-Datenverbindung sein sollte. Dieser wird benötigt, wenn sich die Datei nicht im Qlik Sense-Arbeitsverzeichnis befindet. **Beispiel: *'lib://Table Files/'*** Im Legacymodus für die Skripterstellung werden die folgenden Pfadformate ebenfalls unterstützt: • absolut **Beispiel: *c:\data*** • relativ zu der App, die diese Skriptzeile enthält. **Beispiel: *data***
filename	Der Dateiname, einschließlich der Dateierweiterung .qvw oder .qvf.

Beschränkungen:

Sie können **binary** nicht zum Laden von Daten aus einer App in derselben Qlik Sense Enterprise-Bereitstellung verwenden, indem Sie die App-ID referenzieren. Vielmehr können Sie die Daten nur aus einer *.qvf*-Datei laden.

Beispiele

String	Beschreibung
`Binary lib://DataFolder/customer.qvw;`	In diesem Beispiel muss sich die Datei in der Datenverbindung **Ordner** befinden. Das kann beispielsweise ein Ordner sein, den Ihr Administrator auf dem Qlik Sense-Server erstellt. Klicken Sie im Dateneditor auf **Neue Verbindung erstellen** und wählen Sie unter **Dateispeicherorte** die Option **Ordner** aus.
`Binary customer.qvf;`	In diesem Beispiel muss sich die Datei im Qlik Sense Arbeitsverzeichnis befinden.
`Binary c:\qv\customer.qvw;`	Dieses Beispiel mit einem absoluten Dateipfad funktioniert nur im Legacymodus für die Skripterstellung.

Comment field

Durch diesen Befehl können Sie Kommentare zu Feldern (Metadaten) selbst eingeben oder aus Datenbanken und Tabellen einlesen. Feldnamen, die in der App nicht vorkommen, werden dabei ignoriert. Wenn ein Feldname mehrmals vorkommt, wird der letzte Wert verwendet.

Syntax:

```
comment [fields] *fieldlist using mapname
```

```
comment [field] fieldname with comment
```

Die Mapping-Tabelle mapname muss zwei Spalten enthalten. Die erste Spalte enthält die Feldnamen, die zweite die Kommentare.

Argumente:

Argumente

Argument	Beschreibung
**fieldlist*	Eine durch Komma getrennte Liste der zu kommentierenden Felder. Das Sternchen * steht für alle Felder. Die Wildcards * und ? sind in Feldnamen zugelassen. Beim Gebrauch von Wildcards innerhalb von Feldnamen müssen diese gegebenenfalls in Anführungszeichen stehen.
mapname	Der Name einer Mapping-Tabelle, die durch die Mapping-Befehle **LOAD** oder **SELECT** geladen wurde.
fieldname	Der Name des Feldes, zu dem ein Kommentar eingegeben wird.
comment	Der Kommentar zu dem Feld.

Example 1:

```
commentmap:

mapping LOAD * inline [

a,b

Alpha,This field contains text values

Num,This field contains numeric values

];

comment fields using commentmap;
```

Example 2:

```
comment field Alpha with AFieldContainingCharacters;
```

```
comment field Num with '*A field containing numbers';

comment Gamma with 'Mickey Mouse field';
```

Comment table

Durch diesen Befehl können Sie Kommentare zu Tabellen (Metadaten) selbst eingeben oder aus Datenbanken und Tabellen einlesen.

Tabellennamen, die in der App nicht vorkommen, werden ignoriert. Wenn ein Tabellenname mehrmals vorkommt, wird der letzte Wert verwendet. Zum Einlesen von Kommentaren benutzen Sie das Schlüsselwort.

Syntax:

```
comment [tables] tablelist using mapname
comment [table] tablename with comment
```

Argumente:

Argumente

Argument	Beschreibung
tablelist	(table{,table})
mapname	Der Name einer Mapping-Tabelle, die durch die Mapping-Befehle **LOAD** oder **SELECT** geladen wurde.
tablename	Der Name der Tabelle, zu der ein Kommentar eingegeben wird.
comment	Der Kommentar zu der Tabelle.

Example 1:

```
Commentmap:
mapping LOAD * inline [
a,b
Main,This is the fact table
Currencies, Currency helper table
];
comment tables using Commentmap;
```

Example 2:

```
comment table Main with 'Main fact table';
```

Connect

Der Befehl **CONNECT** legt den Qlik Sense-Zugriff auf eine allgemeine Datenbank über die OLE DB/ODBC-Schnittstelle fest. Für ODBC muss die Datenquelle zunächst mithilfe des ODBC-Administrators angegeben werden.

Diese Funktion ist in Qlik Sense SaaS nicht verfügbar.

Dieser Befehl unterstützt im Standardmodus nur Ordner-Datenverbindungen.

Syntax:

```
ODBC CONNECT TO connect-string
OLEDB CONNECT TO connect-string
CUSTOM CONNECT TO connect-string
LIB CONNECT TO connection
```

Argumente:

Argumente

Argument	Beschreibung

connect-string	`connect-string ::= datasourcename { ; conn-spec-item }`
	Die Datenquelle besteht aus dem Namen der Datenbank und optional einer Reihe von Parametern, welche die Verbindung charakterisieren. Enthält der Name der Datenquelle Leerzeichen oder sind ein oder mehrere Parameter angegeben, muss der Verbindungsstring in Anführungszeichen stehen.
	datasourcename muss eine definierte ODBC-Datenquelle oder ein String sein, der einen OLE DB-Provider angibt.
	`conn-spec-item ::=`**`DBQ=`**`database_specifier |`**`DriverID=`**`driver_specifier |`**`UID=`**`userid |`**`PWD=`**`password`
	Diese Parameter können je nach Datenbank unterschiedlich sein. Für manche Datenbanken sind auch andere als die genannten Parameter denkbar. Für OLE DB sind einige der verbindungsspezifischen Elemente obligatorisch und nicht optional.

connection	Der Name einer im Dateneditor gespeicherten Datenverbindung.

Der Zusatz **ODBC** vor dem **CONNECT**-Befehl bewirkt, dass die Verbindung zur Datenquelle über ODBC statt über OLE DB erfolgt.

LIB CONNECT TO stellt eine Verbindung zu einer Datenbank über eine gespeicherte Datenverbindung her, die im Dateneditor erstellt wurde.

Example 1:

```
ODBC CONNECT TO 'Sales
DBQ=C:\Program Files\Access\Samples\Sales.mdb';
```

Die definierte Datenquelle wird in jedem nachfolgenden **Select (SQL)**-Befehl solange weiter benutzt, bis ein neuer **CONNECT**-Befehl erscheint.

Example 2:

```
LIB CONNECT TO 'DataConnection';
```

Connect32

Dieser Befehl hat dieselbe Funktion wie der **CONNECT**-Befehl, erzwingt jedoch in 64-Bit-Systemen die Verwendung eines 32-Bit-ODBC/OLE DB-Providers. Dieser Befehl ist nicht in Verbindung mit benutzerdefinierten Datenquellen verwendbar.

Connect64

Dieser Befehl hat dieselbe Funktion wie der **CONNECT**-Befehl, erzwingt jedoch die Verwendung eines 64-Bit-Providers. Dieser Befehl ist nicht in Verbindung mit benutzerdefinierten Datenquellen verwendbar.

Declare

Der Befehl **Declare** wird verwendet, um Felddefinitionen zu erstellen, in denen Sie Beziehungen zwischen Feldern und Funktionen festlegen können. Mithilfe mehrerer Felddefinitionen lassen sich automatisch abgeleitete Felder erstellen, die als Dimensionen verwendet werden können. Sie können beispielsweise eine Kalenderdefinition erstellen und diese zur Generierung von damit in Bezug stehenden Dimensionen, wie Jahr, Monat, Woche und Tag, aus einem Datumsfeld verwenden.

Sie können **Declare** verwenden, um entweder eine neue Felddefinition festzulegen oder eine Felddefinition auf der Grundlage einer bereits vorhandenen Definition erstellen.

Festlegen einer neuen Felddefinition

Syntax:

```
definition_name:

Declare [Field[s]] Definition [Tagged tag_list ]

[Parameters parameter_list ]

Fields field_list
```

Argumente:

Argument	Beschreibung
definition_name	Name der Felddefinition, beendet mit einem Doppelpunkt. *Verwenden Sie autoCalendar nicht als Namen für Felddefinitionen, da dieser Name für automatisch generierte Kalendervorlagen reserviert ist.* **Beispiel:** `Calendar:`

Argument	Beschreibung
tag_list	Eine durch Kommas getrennte Liste mit Tags, die auf über die Felddefinition abgeleitete Felder anzuwenden sind. Das Anwenden von Tags ist optional, aber wenn Sie keine Tags wie $date, $numeric oder $text zum Festlegen einer Sortierreihenfolge anwenden, wird das abgeleitete Feld standardmäßig nach Lade-Reihenfolge sortiert. **Beispiel:** `'$date'Thank you for bringing this to our attention, and apologies for the inconvenience.`
parameter_list	Eine durch Kommas getrennte Liste der Parameter. Ein Parameter wird in der Form `name=value` definiert. Ihm wird ein Startwert zugewiesen, der bei der Wiederverwendung der Felddefinition überschrieben werden kann. Optional. **Beispiel:** `first_month_of_year = 1`
field_list	Eine durch Kommas getrennte Liste der zu generierenden Felder, wenn die Felddefinition verwendet wird. Ein Feld wird in der Form `<expression> As field_name tagged tag` definiert. Verwenden Sie `$1` zur Bezugnahme auf das Datenfeld, aus dem die abgeleiteten Felder generiert werden sollen. **Beispiel:** `Year($1) As Year tagged ('$numeric')`

Beispiel:

```
Calendar:
DECLARE FIELD DEFINITION TAGGED '$date'
   Parameters
      first_month_of_year = 1
   Fields

      Year($1) As Year Tagged ('$numeric'),
      Month($1) as Month Tagged ('$numeric'),
      Date($1) as Date Tagged ('$date'),
      Week($1) as Week Tagged ('$numeric'),
      Weekday($1) as Weekday Tagged ('$numeric'),
      DayNumberOfYear($1, first_month_of_year) as DayNumberOfYear Tagged ('$numeric')
;
```

Der Kalender ist jetzt definiert und Sie können ihn auf die Datumsfelder anwenden, die Sie geladen haben, in diesem Fall OrderDate und ShippingDate. Dazu dient eine **Derive**-Bedingung.

Wiederverwenden einer vorhandenen Felddefinition

Syntax:

```
<definition name>:

Declare [Field][s] Definition

Using <existing_definition>

[With <parameter_assignment> ]
```

Argumente:

Argument	Beschreibung
definition_name	Name der Felddefinition, beendet mit einem Doppelpunkt. **Beispiel:** `MyCalendar:`
existing_definition	Die Felddefinition, die beim Erstellen einer neuen Felddefinition wiederverwendet werden soll. Die neue Felddefinition funktioniert wie die Definition, die ihr zugrunde liegt, mit der Ausnahme, wenn parameter_assignment zum Ändern eines Werts in den Feldformeln verwendet wird. **Beispiel:** `Using Calendar`
parameter_assignment	Eine durch Kommas getrennte Liste der Parameterzuweisungen. Eine Parameterzuweisung wird in der Form `name=value` festgelegt und überschreibt den Parameterwert, der in der Basisfelddefinition festgelegt ist. Optional. **Beispiel:** `first_month_of_year = 4`

Beispiel:

In diesem Beispiel verwenden wir die Kalenderdefinition, die im vorhergehenden Beispiel erstellt wurde, erneut. In diesem Fall soll das Geschäftsjahr im April beginnen. Dies wird erreicht, indem dem Parameter first_month_of_year der Wert 4 zugewiesen wird, was sich auf das definierte Feld DayNumberOfYear auswirkt.

Im Beispiel wird angenommen, dass Sie die Beispieldaten und die Felddefinition des vorherigen Beispiels verwenden.

```
MyCalendar:
DECLARE FIELD DEFINITION USING Calendar WITH first_month_of_year=4;
```

```
DERIVE FIELDS FROM FIELDS OrderDate,ShippingDate USING MyCalendar;
```

Wenn Sie das Datenskript neu geladen haben, sind die generierten Felder im Arbeitsblatteditor verfügbar und tragen die Namen OrderDate.MyCalendar.* und ShippingDate.MyCalendar.*.

Derive

Der Befehl **Derive** wird zur Generierung abgeleiteter Felder auf Grundlage einer Felddefinition verwendet, die mit einem **Declare**-Befehl erstellt wurde. Sie können entweder angeben, für welche Datenfelder die Felder abgeleitet werden sollen, oder die Felder explizit oder implizit auf Grundlage der Tags für die Felder ableiten.

Syntax:

```
Derive [fields]] From [Field[s]] field_list Using definition

Derive [Field[s]] From Explicit [Tag[s]] tag_list Using definition

Derive [Field[s]] From Implicit [Tag[s]] Using definition
```

Argumente:

Argumente

Argument	Beschreibung
definition	Name der Felddefinition, der beim Ableiten der Felder verwendet werden soll. **Beispiel:** `Calendar`
field_list	Eine durch Kommas getrennte Liste mit Datenfeldern, aus denen die abgeleiteten Felder auf Grundlage der Felddefinition generiert werden sollen. Bei den Datenfeldern sollte es sich um die Felder handeln, die bereits im Skript geladen sind. **Beispiel:** `OrderDate, ShippingDate`
tag_list	Eine durch Kommas getrennte Liste der Tags. Abgeleitete Felder werden für alle Datenfelder mit beliebigen der aufgelisteten Tags generiert. Die Liste der Tags muss in runden Klammern eingeschlossen werden. **Beispiel:** `('$date','$timestamp')`

Beispiele:

- Ableiten von Feldern für spezifische Datenfelder.
  In diesem Fall legen wir die Felder OrderDate und ShippingDate fest.
  `DERIVE FIELDS FROM FIELDS OrderDate,ShippingDate USING Calendar;`
- Ableiten von Feldern für alle Felder mit einem bestimmten Tag.
  In diesem Fall leiten wir Felder auf der Grundlage von Calendar für alle Felder mit einem `$date`-Tag ab.
  `DERIVE FIELDS FROM EXPLICIT TAGS ('$date') USING Calendar;`
- Ableiten von Feldern für alle Felder mit einem Felddefinitions-Tag.

In diesem Fall leiten wir Felder für alle Datenfelder mit demselben Tag als Calendar-Felddefinition ab, in diesem Fall ist dies `$date`.

```
DERIVE FIELDS FROM IMPLICIT TAG USING Calendar;
```

Direct Query

Mit dem Befehl **DIRECT QUERY** können Sie Tabellen über eine ODBC- oder OLE DB-Verbindung aufrufen, und zwar mithilfe der Funktion Direct Discovery.

Syntax:

```
DIRECT QUERY DIMENSION fieldlist [MEASURE fieldlist] [DETAIL fieldlist] FROM
tablelist
[WHERE where_clause]
```

Die Schlüsselwörter **DIMENSION**, **MEASURE** und **DETAIL** können in beliebiger Reihenfolge verwendet werden.

Die **DIMENSION**- und **FROM**-Schlüsselwortbedingungen sind bei allen **DIRECT QUERY**-Befehlen erforderlich. Das Schlüsselwort **FROM** muss nach dem Schlüsselwort **DIMENSION** stehen.

Die direkt hinter dem Schlüsselwort **DIMENSION** angegebenen Felder werden in den Speicher geladen und können zum Erstellen von Verknüpfungen zwischen im Speicher befindlichen und Direct Discovery-Daten verwendet werden.

*Der Befehl **DIRECT QUERY** kann keine **DISTINCT**- oder **GROUP BY**-Bedingungen enthalten.*

Mithilfe des Schlüsselworts **MEASURE** können Sie Felder definieren, die Qlik Sense auf "Metaebene" erkennt. Die tatsächlichen Daten eines measure-Feldes sind nur während des Ladens der Daten in der Datenbank gespeichert und werden von den Diagrammfeldern, die in einer Visualisierung verwendet werden, ad hoc abgerufen.

Üblicherweise sollten Felder mit diskreten Werten, die als Dimensionen verwendet werden, mit dem Schlüsselwort **DIMENSION** geladen werden, wohingegen Zahlen, die nur in Aggregierungen verwendet werden, mit dem Schlüsselwort **MEASURE** ausgewählt werden sollten.

DETAIL-Felder beinhalten Informationen oder Details, wie z. B. Kommentarfelder, die Benutzern die detailliertere Anzeige von Daten in einer Tabellenbox ermöglichen. **DETAIL**-Felder können nicht als Diagrammformeln verwendet werden.

Der Befehl **DIRECT QUERY** ist für Datenquellen, die SQL unterstützen, datenquellenneutral. Deshalb kann derselbe **DIRECT QUERY**-Befehl für unterschiedliche SQL-Datenbanken unverändert verwendet werden. Direct Discovery generiert bei Bedarf datenbankgeeignete Abfragen.

Eine native Datenquellsyntax kann verwendet werden, wenn der Benutzer weiß, welche Datenbank abgefragt wird und datenbankspezifische Erweiterungen auf SQL anwenden möchte. Eine native Datenquellsyntax wird unterstützt:

- Als Feldformel in **DIMENSION**- und **MEASURE**-Bedingungen
- Als Inhalt der **WHERE**-Bedingung

Beispiele:

```
DIRECT QUERY
      DIMENSION Dim1, Dim2
      MEASURE
            NATIVE ('X % Y') AS X_MOD_Y
FROM TableName
DIRECT QUERY
      DIMENSION Dim1, Dim2
      MEASURE X, Y
      FROM TableName
      WHERE NATIVE ('EMAIL MATCHES "\*.EDU"')
```

Die folgenden Begriffe werden als Schlüsselwörter verwendet und können demnach nicht als Spalten- oder Feldnamen verwendet werden, ohne in Anführungszeichen gesetzt zu werden: and, as, detach, detail, dimension, distinct, from, in, is, like, measure, native, not, or, where

Argumente:

Argument	Beschreibung
fieldlist	Eine kommagetrennte Liste von Feldspezifikationen, *fieldname {, fieldname}*. Eine Feldspezifikation kann ein Feldname sein: In diesem Fall wird derselbe Name für den Datenbankspaltennamen und den Qlik Sense-Feldnamen verwendet. Oder eine Feldspezifikation kann ein Feldalias sein: In diesem Fall erhält eine Datenbankformel oder ein Spaltenname einen Qlik Sense-Feldnamen.
tablelist	Eine Liste der Namen von Tabellen oder Ansichten in der Datenbank, aus denen Daten geladen werden. Üblicherweise sind dies Ansichten, die einen JOIN enthalten, der auf die Datenbank angewendet wurde.

where_ clause	Die vollständige Syntax der **WHERE**-Klauseln der Datenbank wird hier nicht definiert, jedoch sind die meisten „relationalen Formeln“ für SQL zulässig, darunter Funktionsaufrufe, der **LIKE**-Operator für Strings, **IS NULL** und **IS NOT NULL** sowie **IN. BETWEEN** ist nicht eingeschlossen.
	
	NOT ist ein einwertiger Operator, im Gegensatz zum Modifikator für bestimmte Schlüsselwörter.
	
	Beispiele:
	
	`WHERE x > 100 AND "Region Code" IN ('south', 'west')`
	`WHERE Code IS NOT NULL and Code LIKE '%prospect'`
	`WHERE NOT X in (1,2,3)`
	Das letzte Beispiel kann nicht wie folgt geschrieben werden:
	
	`WHERE X NOT in (1,2,3)`

Beispiel:

In diesem Beispiel wird eine Datenbanktabelle mit dem Namen TableName verwendet, die die Felder Dim1, Dim2, Num1, Num2 und Num3 enthält.Dim1 und Dim2 werden in den Qlik Sense Datensatz geladen.

```
DIRECT QUERY DIMENSTION Dim1, Dim2 MEASURE Num1, Num2, Num3 FROM TableName ;
```

Dim1 und Dim2 stehen zur Verwendung als Dimensionen zur Verfügung. Num1, Num2 und Num3 stehen für Aggregierungen zur Verfügung. Dim1 and Dim2 stehen ebenfalls für Aggregierungen zur Verfügung. Die Art der Aggregierungen, für dieDim1 und Dim2 verwendet werden können, richtet sich nach ihrem Datentyp. In vielen Fällen enthalten **DIMENSION**-Felder z. B. String-Daten wie Namen oder Kundennummern. Diese Felder können nicht aggregiert, aber gezählt werden: `count(Dim1)`.

***DIRECT QUERY**-Befehle werden direkt in den Skript-Editor geschrieben. Um den Aufbau von **DIRECT QUERY**-Befehlen zu vereinfachen, können Sie einen **SELECT**-Befehl aus einer Datenverbindung generieren und das generierte Skript anschließend entsprechend einem **DIRECT QUERY**-Befehl abändern.*
*Beispielsweise kann der **SELECT**-Befehl*

```
SQL SELECT
 SalesOrderID,
 RevisionNumber,
 OrderDate,
 SubTotal,
 TaxAmt
FROM MyDB.Sales.SalesOrderHeader;
```

*zu folgendem **DIRECT QUERY**-Befehl geändert werden:*

```
DIRECT QUERY
 DIMENSION
 SalesOrderID,
 RevisionNumber

MEASURE
 SubTotal,
 TaxAmt

DETAIL
OrderDate

FROM MyDB.Sales.SalesOrderHeader;
```

Direct Discovery-Feldlisten

Eine Feldliste ist eine kommagetrennte Liste von Feldspezifikationen, *fieldname {, fieldname}*. Eine Feldspezifikation kann ein Feldname sein: In diesem Fall wird derselbe Name für den Datenbankspaltennamen und den Feldnamen verwendet. Oder eine Feldspezifikation kann ein Feldalias sein: In diesem Fall erhält eine Datenbankformel oder ein Spaltenname einen Qlik Sense-Feldnamen.

Feldnamen können einfache Namen sein oder in Anführungszeichen gesetzt werden. Ein einfacher Name beginnt mit einem alphabetischen Unicode-Zeichen, gefolgt von einer beliebigen Kombination aus alphabetischen oder numerischen Zeichen oder Unterstrichen. Namen in Anführungszeichen beginnen mit einem doppelten Anführungszeichen und können eine beliebige Zeichenfolge enthalten. Wenn ein Name in Anführungszeichen doppelte Anführungszeichen enthält, werden diese durch zwei aufeinanderfolgende doppelte Anführungszeichen dargestellt.

Bei Qlik Sense-Feldnamen wird zwischen Groß- und Kleinschreibung unterschieden. Bei Datenbankfeldnamen wird je nach Datenbank zwischen Groß- und Kleinschreibung unterschieden oder nicht. Bei einer Direct Discovery-Abfrage wird die Groß- bzw. Kleinschreibung aller Feldidentifikatoren und Aliase beibehalten. Im nachfolgenden Beispiel wird der Alias "MyState" intern für die Speicherung der Daten aus der Datenbankspalte "STATEID" verwendet.

```
DIRECT QUERY Dimension STATEID as MyState Measure AMOUNT from SALES_TABLE;
```

Das unterscheidet sich vom Ergebnis eines **SQL Select**-Befehls mit einem Alias. Wenn der Alias nicht explizit in Anführungszeichen gesetzt wird, enthält das Ergebnis die durch die Zieldatenbank ausgegebene Standardschreibung der Spalte. Im folgenden Beispiel erzeugt der **SQL Select**-Befehl an eine Oracle-Datenbank "MYSTATE," als internen Qlik Sense-Alias in Großbuchstaben, obwohl der Alias in gemischter Schreibung angegeben wurde. Der **SQL Select**-Befehl greift auf den Spaltennamen zurück, der von der Datenbank ausgegeben wurde, im Fall von Oracle also Großbuchstaben.

```
SQL Select STATEID as MyState, STATENAME from STATE_TABLE;
```

Um dies zu vermeiden, verwenden Sie den LOAD-Befehl, um den Alias festzulegen.

```
Load STATEID as MyState, STATENAME;
SQL Select STATEID, STATEMENT from STATE_TABLE;
```

In diesem Beispiel wird die "STATEID"-Spalte intern von Qlik Sense als "MyState" gespeichert.

Die meisten skalaren Formeln in Datenbanken sind als Feldspezifikationen zulässig. In Feldspezifikationen können auch Funktionsaufrufe verwendet werden. Formeln können boolesche, numerische Konstanten oder Konstanten in Form von Strings in einfachen Anführungszeichen enthalten (eingebettete einfache Anführungszeichen werden durch danebenliegende einfache Anführungszeichen dargestellt).

Beispiele:

```
DIRECT QUERY

   DIMENSION
```

```
            SalesOrderID, RevisionNumber

         MEASURE

            SubTotal AS "Sub Total"

      FROM AdventureWorks.Sales.SalesOrderHeader;


      DIRECT QUERY

         DIMENSION

            "SalesOrderID" AS "Sales Order ID"

         MEASURE

            SubTotal,TaxAmt,(SubTotal-TaxAmt) AS "Net Total"

      FROM AdventureWorks.Sales.SalesOrderHeader;


      DIRECT QUERY

         DIMENSION

            (2*Radius*3.14159) AS Circumference,

            Molecules/6.02e23 AS Moles

         MEASURE

            Num1 AS numA

      FROM TableName;


DIRECT QUERY
   DIMENSION
      concat(region, 'code') AS region_code
   MEASURE
      Num1 AS NumA
FROM TableName;
```

Direct Discovery unterstützt nicht die Verwendung von Aggregierungen in **LOAD**-Befehlen. Bei der Verwendung von Aggregierungen sind die Ergebnisse unvorhersehbar. **LOAD**-Befehle wie die folgenden sollten nicht verwendet werden:

```
DIRECT QUERY DIMENSION stateid, SUM(amount*7) AS MultiFirst MEASURE amount FROM sales_table;
```

Die **SUM** sollte nicht in den **LOAD**-Befehl eingebunden werden.

Direct Discovery unterstützt auch nicht die Verwendung von Qlik Sense-Funktionen in **Direct Query**-Befehlen. Beispielsweise führt die folgende Spezifikation eines **DIMENSION**-Feldes zum fehlerhaften Ergebnis, wenn das Feld "Mth" als Dimension in einer Visualisierung verwendet wird:

```
month(ModifiedDate) as Mth
```

Directory

Die **Directory**-Anweisung definiert, welches Verzeichnis in den nachfolgenden **LOAD**-Anweisungen nach Datendateien durchsucht wird, bis eine neue **Directory**-Anweisung erstellt wird.

Syntax:

```
Directory[path]
```

Wenn der **Directory**-Befehl ohne **path** ausgegeben oder ausgelassen wird, sucht Qlik Sense im Qlik Sense-Arbeitsverzeichnis.

Argumente:

Argumente

Argument	Beschreibung
path	Ein Text, der als Pfad zur data-Datei interpretiert werden kann. Der Pfad bezeichnet den Pfad zu der Datei auf eine der folgenden Weisen: • absolut **Beispiel: *c:\data*** • relativ zum Qlik Sense-App-Arbeitsverzeichnis. **Beispiel: *data*** • als URL-Adresse (HTTP oder FTP), die eine Datei im Internet oder Intranet lokalisiert. **Beispiel: *http://www.qlik.com***

Beispiele:

```
DIRECTORY C:\userfiles\data; // OR -> DIRECTORY data\

LOAD * FROM
[data1.csv] // ONLY THE FILE NAME CAN BE SPECIFIED HERE (WITHOUT THE FULL PATH)
(ansi, txt, delimiter is ',', embedded labels);

LOAD * FROM
[data2.txt] // ONLY THE FILE NAME CAN BE SPECIFIED HERE UNTIL A NEW DIRECTORY STATEMENT IS
MADE
(ansi, txt, delimiter is '\t', embedded labels);
```

Disconnect

Der **Disconnect**-Befehl beendet die aktuelle ODBC/OLE DB/benutzerdefinierte Verbindung. Der Befehl ist optional.

Syntax:

```
Disconnect
```

Die Verbindung wird automatisch beendet, wenn ein neuer **connect**-Befehl im Skript erscheint oder wenn die Ausführung des Skripts beendet ist.

Beispiel:

```
Disconnect;
```

Drop

Das Skriptschlüsselwort **Drop** kann zum Ersetzen von Tabellen oder Feldern in einer Datenbank verwendet werden.

Drop field

Durch den Befehl **drop field** können jederzeit ein oder mehrere Qlik Sense-Felder während der Skriptausführung aus dem Datenmodell und damit aus dem Arbeitsspeicher gelöscht werden. Die Eigenschaft „distinct“ einer Tabelle wird nach einem **drop field**-Befehl entfernt.

*Sowohl **drop field** als auch **drop fields** ist zulässig. Die Funktion der beiden Befehle ist identisch. Ist kein Tabellenname angegeben, wird das Feld aus allen Tabellen gelöscht, in denen es vorkommt.*

Syntax:

```
Drop field fieldname { , fieldname2 ...} [from tablename1 { , tablename2
...}]
Drop fields fieldname { , fieldname2 ...} [from tablename1 { , tablename2
...}]
```

Beispiele:

```
Drop field A;
Drop fields A,B;
Drop field A from X;
Drop fields A,B from X,Y;
```

Drop table

Durch den Befehl **drop table** können jederzeit eine oder mehrere interne Qlik Sense-Tabellen während der Skriptausführung aus dem Datenmodell und damit aus dem Arbeitsspeicher gelöscht werden.

Syntax:

```
drop table  tablename {, tablename2 ...}
drop tables tablename {, tablename2 ...}
```

*Es werden die Formulare **drop table** und **drop tables** akzeptiert.*

Bei diesem Vorgang werden folgende Informationen gelöscht:

- die Tabelle(n)
- die Felder, die ausschließlich in der zu löschenden Tabelle vorkommen
- die Werte verbleibender Felder, die ausschließlich in der zu löschenden Tabelle vorkommen

Beispiele und Ergebnisse:

Beispiel	**Ergebnis**
`drop table Orders, Salesmen, T456a;`	Durch diesen Befehl werden die drei genannten Tabellen gelöscht.

`Tab1:`
`Load * Inline [`
`Customer, Items, UnitPrice`
`Bob, 5, 1.50`
`];`
 	
`Tab2:`
`LOAD Customer, Sum( Items * UnitPrice ) as Sales`
`resident Tab1`
`group by Customer;`
 	
`drop table Tab1;`
	Nachdem die Tabelle *Tab2* erzeugt wurde, wird *Tab1* gelöscht.

Drop table

Durch den Befehl **drop table** können jederzeit eine oder mehrere interne Qlik Sense-Tabellen während der Skriptausführung aus dem Datenmodell und damit aus dem Arbeitsspeicher gelöscht werden.

Syntax:

```
drop table  tablename {, tablename2 ...}
drop tables tablename {, tablename2 ...}
```

*Es werden die Formulare **drop table** und **drop tables** akzeptiert.*

Bei diesem Vorgang werden folgende Informationen gelöscht:

- die Tabelle(n)
- die Felder, die ausschließlich in der zu löschenden Tabelle vorkommen
- die Werte verbleibender Felder, die ausschließlich in der zu löschenden Tabelle vorkommen

Beispiele und Ergebnisse:

Beispiel	Ergebnis
`drop table Orders, Salesmen, T456a;`	Durch diesen Befehl werden die drei genannten Tabellen gelöscht.

`Tab1:`
`Load * Inline [`
`Customer, Items, UnitPrice`
`Bob, 5, 1.50`
`];`
 	
`Tab2:`
`LOAD Customer, Sum( Items * UnitPrice ) as Sales`
`resident Tab1`
`group by Customer;`
 	
`drop table Tab1;`
	Nachdem die Tabelle *Tab2* erzeugt wurde, wird *Tab1* gelöscht.

Execute

Der Befehl **Execute** wird zur Ausführung anderer Programme verwendet, während Qlik Sense Daten lädt. Dies dient z. B. dazu, notwendige Konvertierungen vorzunehmen.

Diese Funktion ist in Qlik Sense SaaS nicht verfügbar.

Dieser Befehl wird im Standardmodus nicht unterstützt.

Syntax:

```
execute commandline
```

Argumente:

Argumente

Argument	Beschreibung
commandline	Ein Text, der vom Betriebssystem als Befehlszeile interpretiert werden kann. Sie können sich auf einen absoluten Dateipfad oder einen lib://-Ordnerpfad beziehen.

Wenn Sie **Execute** verwenden möchten, müssen folgende Voraussetzungen erfüllt sein:

- Der Legacymodus muss ausgeführt werden (gilt für Qlik Sense und Qlik Sense Desktop).
- Sie müssen OverrideScriptSecurity in *Settings.ini* auf 1 setzen (gilt für Qlik Sense). *Settings.ini* befindet sich unter *C:\ProgramData\Qlik\Sense\Engine* und ist für gewöhnlich eine leere Datei.

*Stellen Sie OverrideScriptSecurity auf die Aktivierung von **Execute** ein, kann jeder Benutzer Dateien auf dem Server ausführen. Ein Benutzer kann z. B. eine ausführbare Datei an eine App anhängen und die Datei dann im Datenladeskript ausführen.*

Gehen Sie folgendermaßen vor:

1. Kopieren Sie *Settings.ini* und öffnen Sie es im Texteditor.
2. Prüfen Sie, dass die Datei in der ersten Zeile *[Settings 7]* enthält.
3. Fügen Sie eine neue Zeile ein und geben Sie *OverrideScriptSecurity=1* ein.
4. Fügen Sie am Ende der Datei eine leere Zeile ein.
5. Speichern Sie die Datei.
6. Ersetzen Sie *Settings.ini* mit der soeben bearbeiteten Datei.
7. Starten Sie Qlik Sense Engine Service (QES) neu.

Falls Qlik Sense als Dienst ausgeführt wird, werden manche Befehle womöglich nicht wie gewohnt ausgeführt.

Beispiel:

```
Execute C:\Program Files\Office12\Excel.exe;
Execute lib://win\notepad.exe // win is a folder connection referring to c:\windows
```

Field/Fields

Die Skriptschlüsselwörter **Field** und **Fields** werden in den Befehlen **Declare**, **Derive**, **Drop**, **Comment**, **Rename** und **Tag/Untag** verwendet.

FlushLog

Der Befehl **FlushLog** erzwingt, dass Qlik Sense den Inhalt des Skriptpuffers in die Skriptprotokolldatei schreibt.

Syntax:

```
FlushLog
```

Der Buffer-Inhalt wird in die Log-Datei geschrieben. Dieser Befehl eignet sich besonders zum Debugging, da Sie damit Daten erhalten, die ansonsten möglicherweise bei einer fehlgeschlagenen Skriptausführung verloren gehen.

Beispiel:

```
FlushLog;
```

Force

Der Befehl **force** erzwingt, dass Qlik Sense die Feldnamen und Feldwerte der folgenden **LOAD**- und **SELECT**-Befehle als in Großbuchstaben, in Kleinbuchstaben oder wie angegeben (gemischt) interpretiert. Auf diese Weise können Tabellen über Schlüsselfelder automatisch miteinander verknüpft werden, auch wenn die Feldnamen in den Datenquellen hinsichtlich Groß- und Kleinschreibung nicht übereinstimmen.

Syntax:

```
Force ( capitalization | case upper | case lower | case mixed )
```

Wird kein force-Befehl benutzt, liest die Software die Daten in der ursprünglichen Schreibweise, also case mixed ein. Der force-Befehl bleibt gültig, bis ein neuer force-Befehl erscheint.

Der **force**-Befehl hat keinerlei Einfluss auf die Zugriffskontrolle im Zugriffsabschnitt, weil alle geladenen Feldwerte unabhängig von Groß- und Kleinschreibung sind.

Beispiele und Ergebnisse

Beispiel	**Ergebnis**

Dieses Beispiel zeigt, wie die Großschreibung erzwungen wird:
`FORCE Capitalization;`
`Capitalization:`
`LOAD * Inline [`
`ab`
`Cd`
`eF`
`GH`
`];`
	Die Tabelle **Capitalization** enthält die folgenden Werte:
	`Ab`
	`Cd`
	`Ef`
	`Gh`
	Alle Werte werden großgeschrieben.

Beispiel	Ergebnis

Dieses Beispiel zeigt, wie die Großschreibung erzwungen wird:
`FORCE Case Upper;`
`CaseUpper:`
`LOAD * Inline [`
`ab`
`Cd`
`eF`
`GH`
`];`
	Die Tabelle **CaseUpper** enthält die folgenden Werte:
	`AB`
	`CD`
	`EF`
	`GH`
	Alle Werte beginnen mit einem Großbuchstaben.

Dieses Beispiel zeigt, wie die Kleinschreibung erzwungen wird:
`FORCE Case Lower;`
`CaseLower:`
`LOAD * Inline [`
`ab`
`Cd`
`eF`
`GH`
`];`
	Die Tabelle **CaseLower** enthält die folgenden Werte:
	`ab`
	`cd`
	`ef`
	`gh`
	Alle Werte beginnen mit einem Kleinbuchstaben.

Beispiel	Ergebnis

Dieses Beispiel zeigt, wie eine gemischte Groß-/Kleinschreibung erzwungen wird.
`FORCE Case Mixed;`
`CaseMixed:`
`LOAD * Inline [`
`ab`
`Cd`
`eF`
`GH`
`];`
	Die Tabelle **CaseMixed** enthält die folgenden Werte:
	`ab`
	`Cd`
	`eF`
	`GH`
	Alle Werte entsprechen Ihrer Darstellung im Skript.

Siehe auch:

From

Das Skriptschlüsselwort **From** wird in **Load**-Befehlen zum Verweisen auf eine Datei verwendet und in **Select**-Befehlen zum Verweisen auf eine Datenbanktabelle oder -ansicht.

Load

Der **LOAD**-Befehl lädt Felder aus einer Datei aus Daten, die im Skript definiert sind, aus einer zuvor geladenen Tabelle, aus einer Webseite, aus dem Ergebnis eines nachfolgenden **SELECT**-Befehls oder durch automatisches Generieren der Daten. Daten können auch aus Analyseverbindungen geladen werden.

Syntax:

```
LOAD [ distinct ] fieldlist

[( from file [ format-spec ] |

from_field fieldassource [format-spec]|

inline data [ format-spec ] |

resident table-label |

autogenerate size ) |extension pluginname.functionname([script]
tabledescription)]
```

```
[ where criterion | while criterion ]
```

```
[ group by groupbyfieldlist ]
```

```
[order by orderbyfieldlist ]
```

Argumente:

Argumente

Argument	Beschreibung
distinct	Sie können **distinct** als Prädikat verwenden, wenn Sie nur eindeutige Datensätze laden möchten. Wenn duplizierte Datensätze vorhanden sind, wird die erste Instanz geladen. Wenn Sie vorangehende Ladevorgänge verwenden, müssen Sie **distinct** in den ersten load-Befehl platzieren, da **distinct** nur die Zieltabelle betrifft.

Argument	Beschreibung
fieldlist	*fieldlist ::= (* | field{**,** * | field })* Liste der zu ladenden Felder. * in einer Felderliste bedeutet alle Felder in der Tabelle. *field ::= (fieldref | expression) [***as** *aliasname]* Die Felddefinition muss immer eine Literale enthalten, einen Verweis auf ein bestehendes Feld oder eine Formel. *fieldref ::= (fieldname |@fieldnumber |@startpos:endpos [* ***I*** | ***U*** | ***R*** | ***B*** | **T**]) Dabei ist *fieldname* ein Text, der einem Feldnamen in der Tabelle entspricht. Beachten Sie, dass der Feldname zwischen geraden doppelten Anführungszeichen oder eckigen Klammern stehen muss, wenn er beispielsweise Leerzeichen enthält. Mitunter sind Feldnamen nicht im Tabellenkopf verfügbar. Verwenden Sie in diesem Fall stattdessen folgende Notation: *@fieldnumber* bezeichnet die Nummer des Feldes in einer Textdatei mit Trennzeichen. Es muss eine ganze positive Zahl mit vorangehendem "@" sein. Die Nummerierung beginnt stets mit 1 und endet mit der Gesamtzahl der Felder. *@startpos*:*endpos* bezeichnet Start- und Endposition eines Feldes in einer Datei mit festen Satzlängen. Die Positionen müssen beide positive ganze Zahlen sein. Beiden Zahlen muss ein "@" vorangehen und sie müssen durch einen Doppelpunkt getrennt sein. Die Nummerierung beginnt stets mit 1 und geht bis zu der entsprechenden Zahl von Stellen. Im letzten Feld wird **n** als Endposition verwendet. • Folgt direkt hinter @*startpos*:*endpos* der Buchstabe **I** oder **U**, werden die eingelesenen Daten als binär codierte ganze Zahl mit Vorzeichen (**I**) bzw. als binär codierte ganze Zahl ohne Vorzeichen **U** (Intel Byte Order) interpretiert. Die Zahl der eingelesenen Positionen muss 1, 2 oder 4 betragen. • Folgt direkt hinter @*startpos*:*endpos* der Buchstabe **R**, werden die eingelesenen Daten als binär codierte reelle Zahlen (IEEE 32-Bit- oder 64-Bit-Gleitkommazahl) interpretiert. Die Zahl der eingelesenen Positionen muss 4 oder 8 betragen. • Folgt direkt hinter @*startpos*:*endpos* der Buchstabe **B**, werden die eingelesenen Daten als binär codierte Dezimalzahlen BCD (Binary Coded Decimal) entsprechend dem COMP-3-Standard interpretiert. Es kann eine beliebige Zahl von Bytes angegeben werden. *expression* kann eine numerische Funktion oder eine Stringfunktion sein, die sich auf ein oder mehrere Felder derselben Tabelle bezieht. Weitere Informationen finden Sie in den Erläuterungen der Formel-Syntax. Der Zusatz **as** weist dem Feld einen neuen Namen zu.

Argument	Beschreibung
from	**from** wird benutzt, um Daten aus einer Datei mithilfe einer Ordner- oder Web-Datei-Datenverbindung zu laden. *file ::= [path] filename* **Beispiel: *'lib://Table Files/'*** Ist kein Pfad angegeben, sucht Qlik Sense die Datei in dem Verzeichnis, das im **Directory**-Befehl angegeben ist. Wenn kein **Directory**-Befehl vorhanden ist, sucht Qlik Sense im Arbeitsverzeichnis *C:\Users\{user}\Documents\Qlik\Sense\Apps*. *In einer Qlik Sense-Server-Installation wird das Arbeitsverzeichnis in Qlik Sense Repository Service angegeben, standardmäßig ist dies C:\ProgramData\Qlik\Sense\Apps.* In *filename* können die in DOS üblichen Wildcards verwendet werden (* und ?). Dadurch werden alle passenden Dateien aus dem angegebenen Verzeichnis geladen. *format-spec ::= (fspec-item { , fspec-item })* Die Formatbezeichnung besteht aus einer Auflistung von Formatoptionen, die in Klammern stehen. **Legacymodus für die Skripterstellung** Im Legacymodus für die Skripterstellung werden die folgenden Pfadformate ebenfalls unterstützt: • absolut **Beispiel: *c:\data*** • relativ zum Qlik Sense-App-Arbeitsverzeichnis. **Beispiel: *data*** • als URL-Adresse (HTTP oder FTP), die eine Datei im Internet oder Intranet lokalisiert. **Beispiel: *http://www.qlik.com*** •

Argument	Beschreibung
from_field	Der Zusatz **from_field** wird benutzt, um Daten aus einem bereits geladenen Feld zu laden. *fieldassource::=***(***tablename, fieldname****)*** Das Feld wird durch *tablename* und *fieldname* definiert. *format-spec ::= (fspec-item {, fspec-item })* Die Formatbezeichnung besteht aus einer Auflistung von Formatoptionen, die in Klammern stehen. Weitere Informationen finden Sie unter *Formatoptionen (page 172)*. ***from_field*** *unterstützt nur Kommas als Listentrennzeichen, wenn Felder in Tabellen getrennt werden.*
inline	Der Zusatz **inline** wird benutzt, wenn Daten direkt in das Skript eingegeben und nicht aus einer Datei geladen werden sollen. *data ::= [text]* Daten, die durch eine **inline**-Bedingung in das Skript eingefügt werden, müssen in doppelten Anführungszeichen oder in eckigen Klammern stehen. Der Text wird auf die gleiche Weise interpretiert wie der Inhalt einer Datei, d. h. er sollte auch genauso aufgebaut sein. Wenn Sie beispielsweise in einer Textdatei eine neue Zeile beginnen würden, sollten Sie dies auch im Text eines **inline**-Befehls durch Drücken der ENTER-Taste tun. Die Anzahl der Spalten wird durch die erste Zeile definiert. *format-spec ::= (fspec-item {, fspec-item })* Die Formatbezeichnung besteht aus einer Auflistung von Formatoptionen, die in Klammern stehen. Weitere Informationen finden Sie unter *Formatoptionen (page 172)*.
resident	Der Zusatz **resident** wird benutzt, um Daten aus einer bereits geladenen Tabelle zu laden. *table label* ist die Bezeichnung, die dem **LOAD**- oder **SELECT**-Befehl vorangeht, durch den die Tabelle erstellt wurde. Am Ende des resident-Zusatzes sollte ein Doppelpunkt stehen.
autogenerate	Der Zusatz **autogenerate** wird benutzt, um Daten automatisch durch Qlik Sense generieren zu lassen. *size ::= number* *Number* steht für eine ganze Zahl, die bezeichnet, wie viele Datensätze angelegt werden sollen. Die Feldliste darf keine Formeln enthalten, die Daten aus einer externen Datenquelle oder einer zuvor geladenen Tabelle enthalten, es sei denn, Sie nehmen auf einen einzigen Feldwert in einer zuvor geladenen Tabelle mit der Funktion **Peek** Bezug.

Argument	Beschreibung
extension	Sie können Daten aus Analyseverbindungen laden. Sie müssen die **extension**-Bedingung verwenden, um eine Funktion aufzurufen, die im Plugin für serverseitige Erweiterung (SSE) definiert ist, oder ein Skript auswerten. Sie können eine einzelne Tabelle an das SSE-Plugin senden, worauf eine einzelne Datentabelle zurückgegeben wird. Wenn das Plugin nicht die Namen der zurückzugebenden Felder angibt, werden die Felder Field1, Field2 usw. benannt. `Extension pluginname.functionname( tabledescription );` • Laden von Daten mithilfe einer Funktion in einem SSE-Plugin *tabledescription ::= (table { ,tablefield})* Wenn Sie keine Tabellenfelder angeben, werden die Felder in der Ladereihenfolge verwendet. • Laden von Daten durch Auswertung eines Skripts in einem SSE-Plugin *tabledescription ::= (script, table { ,tablefield})* **Umgang mit Datentypen in der Tabellenfelddefinition** Datentypen werden in Analyseverbindungen automatisch erkannt. Wenn die Daten keinen numerischen Wert und mindestens einen Nicht-NULL-Textstring enthalten, wird das Feld als Text betrachtet. In allen anderen Fällen wird es als numerisch betrachtet. Sie können den Datentyp erzwingen, indem Sie einen Feldnamen mit **String()** oder mit **Mixed()** umgeben. • **String()** erzwingt, dass ein Feld als Text betrachtet wird. Wenn das Feld numerisch ist, wird der Textteil des dualen Werts extrahiert und keine Konvertierung vorgenommen. • **Mixed()** erzwingt, dass das Feld als dual betrachtet wird. **String()** oder **Mixed()** können nicht außerhalb von **extension**-Tabellenfelddefinitionen verwendet werden, und Sie können keine anderen Qlik Sense Funktionen in einer Tabellenfelddefinition verwenden. **Weitere Informationen zu Analyseverbindungen** Sie müssen Analyseverbindungen konfigurieren, bevor Sie sie verwenden können.
where	**where** wird benutzt, um anhand eines Kriteriums zu prüfen, ob der Datensatz geladen wird oder nicht. Ist *criterion* True, wird der Datensatz geladen. *criterion* ist eine logische Formel.
while	Der Zusatz **while** dient dazu, einen Datensatz mehrfach zu laden. Der Datensatz wird solange eingelesen, wie *criterion* True ist. Sinnvollerweise schließt ein **while**-Zusatz die Funktion **IterNo( )** ein. *criterion* ist eine logische Formel.

Argument	Beschreibung
group by	Der Zusatz **group by** bestimmt, nach welchen Feldern die Daten aggregiert (gruppiert) werden sollen. Die Aggregierungsfelder sollten in den geladenen Formeln verwendet werden. Außerhalb von Aggregierungsfunktionen dürfen in load-Befehlen mit group by-Zusatz nur die im group by-Zusatz aufgeführten Felder geladen werden. *groupbyfieldlist ::= (fieldname { ,fieldname })*
order by	Der Zusatz **order by** kann nur in **load**-Befehlen mit resident-Zusatz benutzt werden, d. h. wenn Daten aus einer bereits geladenen Tabelle eingelesen werden. Er dient dazu, die Datensätze der im resident-Zusatz bezeichneten Tabelle vor dem erneuten Einlesen zu sortieren. Dabei kann nach einem oder mehreren Felder in auf- oder absteigender Reihenfolge sortiert werden. Die Sortierung erfolgt zunächst nach numerischen Werten, dann nach nationaler Sortierreihenfolge. Diese Bedingung darf nur verwendet werden, wenn die Datenquelle eine bezeichnete Tabelle ist. Mit den Ordnungsfeldern wird angegeben, nach welchem Feld die bezeichnete Tabelle sortiert wird. Das Feld kann durch den Feldnamen oder die Feldnummer bezeichnet sein (die Nummerierung beginnt stets bei 1). orderbyfieldlist ::= fieldname [sortorder] { , fieldname [sortorder] } Hinter *sortorder* folgt entweder *asc* für aufsteigend oder *desc* für absteigend. Fehlt die *sortorder*, wird *asc* angenommen. *fieldname*, *path*, *filename* und *aliasname* sind Strings. Als *fieldname* kann jedes Feld in der Quelltabelle verwendet werden. Felder, die mithilfe der as-Bedingung (*aliasname*) erstellt werden, befinden sich außerhalb des Gültigkeitsbereichs und können nicht im gleichen **load**-Befehl verwendet werden.

Wenn durch keinen der Zusätze **from**, **inline**, **resident from_field** oder **autogenerate** eine Datenquelle definiert wird, werden Daten aus dem Ergebnis des direkt nachfolgenden **SELECT**- oder **LOAD**-Befehls geladen. Diesem nachfolgenden Befehl sollte kein Zusatz vorangehen.

Beispiele:

Laden verschiedener Dateiformate
Laden einer Datei mit Trennzeichen mit Standardoptionen:

```
LOAD * from data1.csv;
```

Laden einer Datei mit Trennzeichen aus einer Bibliotheksverbindung (DataFiles):

```
LOAD * from 'lib://DataFiles/data1.csv';
```

Laden aller Dateien mit Trennzeichen aus einer Bibliotheksverbindung (DataFiles):

```
LOAD * from 'lib://DataFiles/*.csv';
```

Laden einer Datei mit Trennzeichen, mit Komma als Trennzeichen und mit der Formatoption "embedded labels":

```
LOAD * from 'c:\userfiles\data1.csv' (ansi, txt, delimiter is ',', embedded labels);
```

Laden einer Datei mit Trennzeichen, mit Tab als Trennzeichen und mit der Formatoption "embedded labels":

```
LOAD * from 'c:\userfiles\data2.txt' (ansi, txt, delimiter is '\t', embedded labels);
```

Laden einer dif-Datei mit eingebetteten Kopfzeilen:

```
LOAD * from file2.dif (ansi, dif, embedded labels);
```

Laden von drei Feldern aus einer Datei mit festen Satzlängen ohne Kopfzeilen:

```
LOAD @1:2 as ID, @3:25 as Name, @57:80 as City from data4.fix (ansi, fix, no labels, header is
0, record is 80);
```

Laden einer QVX-Datei, mit einem absoluten Pfad:

```
LOAD * from C:\qdssamples\xyz.qvx (qvx);
```

Laden von Webdateien
Laden aus der Standard-URL, die in der Webdatei-Datenverbindung festgelegt ist:

```
LOAD * from [lib://MyWebFile];
```

Laden aus einer spezifischen URL und Überschreiben der URL, die in der Webdatei-Datenverbindung festgelegt ist:

```
LOAD * from [lib://MyWebFile] (URL is 'http://localhost:8000/foo.bar');
```

Laden aus einer spezifischen URL, die in einer Variablen festgelegt ist, mit Aufrufen:

```
SET dynamicURL = 'http://localhost/foo.bar';

LOAD * from [lib://MyWebFile] (URL is '$(dynamicURL)');
```

Auswahl, Umbenennung und Berechnung von Feldern
Laden von nur drei spezifischen Feldern aus einer Datei mit Trennzeichen:

```
LOAD FirstName, LastName, Number from data1.csv;
```

Erstes Feld zu A umbenennen und zweites Feld zu B umbenennen beim Laden einer Datei ohne Bezeichnungen:

```
LOAD @1 as A, @2 as B from data3.txt (ansi, txt, delimiter is '\t', no labels);
```

Laden von Name als Zusammenfassung von FirstName, einem Leerzeichen und LastName:

```
LOAD FirstName&' '&LastName as Name from data1.csv;
```

Laden von Quantity, Price und Value (das Produkt aus Quantity und Price):

```
LOAD Quantity, Price, Quantity*Price as Value from data1.csv;
```

Auswahl von Datensätzen
Nur einzigartige Datensätze laden, doppelte Datensätze werden ausgeschlossen:

```
LOAD distinct FirstName, LastName, Number from data1.csv;
```

Nur Datensätze laden, bei denen das Feld Litres einen Wert über Null hat:

```
LOAD * from Consumption.csv where Litres>0;
```

Laden von direkt in das Skript eingegebenen Daten und autogenerierten Daten
Laden einer Tabelle mit Inline-Daten, zwei Feldern mit den Namen CatID und Category:

```
LOAD * Inline

[CatID, Category

0,Regular

1,Occasional

2,Permanent];
```

Laden einer Tabelle mit Inline-Daten, drei Feldern mit den Namen UserID, Password und Access:

```
LOAD * Inline [UserID, Password, Access

A, ABC456, User

B, VIP789, Admin];
```

Laden einer Tabelle mit 10 000 Zeilen. Feld A enthält die Zahl des Datensatzes (1,2,3,4,5...) und Feld B enthält eine zufällige Zahl zwischen 0 und 1:

```
LOAD RecNo( ) as A, rand( ) as B autogenerate(10000);
```

Die Klammer nach autogenerate ist zulässig, aber nicht erforderlich.

Laden von Daten aus bereits geladenen Tabellen
Zunächst laden wir eine Tabellendatei mit Trennzeichen und benennen sie tab1:

```
tab1:

SELECT A,B,C,D from 'lib://DataFiles/data1.csv';
```

Laden von Feldern aus der bereits geladenen tab1-Tabelle als tab2:

```
tab2:

LOAD A,B,month(C),A*B+D as E resident tab1;
```

Laden von Feldern aus der bereits geladenen Tabelle tab1, aber nur Datensätze bei denen A größer als B ist:

```
tab3:

LOAD A,A+B+C resident tab1 where A>B;
```

Laden von Feldern aus der bereits geladenen Tabelle tab1 sortiert nach A:

```
LOAD A,B*C as E resident tab1 order by A;
```

Laden von Feldern aus der bereits geladenen Tabelle tab1, sortiert nach dem ersten Feld und dann nach dem zweiten Feld:

```
LOAD A,B*C as E resident tab1 order by 1,2;
```

Laden von Feldern aus der bereits geladenen Tabelle tab1 absteigend sortiert nach C, dann aufsteigend nach B und dann nach dem ersten Feld in absteigender Reihenfolge:

```
LOAD A,B*C as E resident tab1 order by C desc, B asc, 1 desc;
```

Laden von Daten aus einem bereits geladenen Feld
Laden des Feldes Types aus der bereits geladenen Tabelle Characters als A:

```
LOAD A from_field (Characters, Types);
```

Laden von Daten aus nachfolgender Tabelle (vorangehender Load-Befehl)
Laden von A, B und berechneten Feldern X und Y aus Table1, die in dem folgenden **SELECT**-Befehl geladen wird:

```
LOAD A, B, if(C>0,'positive','negative') as X, weekday(D) as Y;

SELECT A,B,C,D from Table1;
```

Gruppierung von Daten
Laden von Feldern gruppiert (aggregiert) nach ArtNo:

```
LOAD ArtNo, round(Sum(TransAmount),0.05) as ArtNoTotal from table.csv group by ArtNo;
```

Laden von Feldern gruppiert (aggregiert) nach Week und ArtNo:

```
LOAD Week, ArtNo, round(Avg(TransAmount),0.05) as WeekArtNoAverages from table.csv group by
Week, ArtNo;
```

Mehrfaches Einlesen eines Datensatzes

In diesem Beispiel gibt es eine Inputdatei Grades.csv, welche die Noten aller Schüler zusammengefasst in einem Feld enthält:

```
Student,Grades

Mike,5234

John,3345

Pete,1234

Paul,3352
```

Die Noten zwischen 1 und 5 stehen für die Fächer Math, English, Science und History. Wir können die Noten in separate Werte aufteilen, indem wir jeden Datensatz mehrmals mit einer **while**-Bedingung einlesen, wobei die **IterNo( )**-Funktion als Zähler dient. Bei jedem Lesevorgang wird die Note mit der **Mid**-Funktion extrahiert und in der Grade gespeichert und das Fach wird mit der **pick**-Funktion ausgewählt und im Subject gespeichert. Die letzte **while**-Bedingung enthält den Test zur Überprüfung, ob alle Noten eingelesen wurden (in diesem Fall vier pro Schüler), was bedeutet, dass der nächste Schülerdatensatz gelesen werden sollte.

```
MyTab:

LOAD Student,

mid(Grades,IterNo( ),1) as Grade,

pick(IterNo( ), 'Math', 'English', 'Science', 'History') as Subject from Grades.csv

while IsNum(mid(Grades,IterNo(),1));
```

Das Ergebnis ist eine Tabelle mit folgenden Daten:

Student	Subject	Grade
John	English	3
John	History	5
John	Math	3
John	Science	4
Mike	English	2
Mike	History	4
Mike	Math	5
Mike	Science	3
Paul	English	3
Paul	History	2
Paul	Math	3
Paul	Science	5
Pete	English	2
Pete	History	4
Pete	Math	1
Pete	Science	3

Laden aus Analyseverbindungen
Die folgenden Beispieldaten werden verwendet.

```
Values:
Load
 Rand() as A,
 Rand() as B,
 Rand() as C
AutoGenerate(50);
```

Laden von Daten mithilfe einer Funktion

In diesem Beispiel wird davon ausgegangen, dass ein Analyseverbindungs-Plugin mit dem Namen *P* vorhanden ist, das eine benutzerdefinierte Funktion *Calculate(Parameter1, Parameter2)* enthält. Die Funktion gibt die Tabelle *Results* zurück, die die Felder *Field1* und *Field2* enthält.

```
Load * Extension P.Calculate( Values{A, C} );
```
Alle Felder laden, die zurückgegeben werden, wenn die Felder A und C an die Funktion gesendet werden.

```
Load Field1 Extension P.Calculate( Values{A, C} );
```
Nur das Feld Field1 laden, wenn die Felder A und C an die Funktion gesendet werden.

```
Load * Extension P.Calculate( Values );
```
Alle Felder laden, die zurückgegeben werden, wenn die Felder A und B an die Funktion gesendet werden. Da keine Felder angegeben sind, werden A und B in der Reihenfolge ihres Vorkommens in der Tabelle verwendet.

```
Load * Extension P.Calculate( Values {C, C});
```
Alle Felder laden, die zurückgegeben werden, wenn Feld C an beide Parameter der Funktion gesendet werden.

```
Load * Extension P.Calculate( Values {String(A), Mixed(B)});
```
Alle Felder laden, die zurückgegeben werden, wenn Feld A als String erzwungen und Feld B als numerisch erzwungen an die Funktion gesendet werden.

Laden von Daten durch Auswertung eines Skripts

```
Load A as A_echo, B as B_echo Extension R.ScriptEval( 'q;', Values{A, B} );
```
Die Tabelle laden, die von Skript q zurückgegeben wird, wenn die Werte von A und B gesendet werden.

```
Load * Extension R.ScriptEval( '$(My_R_Script)', Values{A, B} );
```
Die Tabelle laden, die von dem in der Variablen My_R_Script gespeicherten Skript zurückgegeben wird, wenn die Werte von A und B gesendet werden.

```
Load * Extension R.ScriptEval( '$(My_R_Script)', Values{B as D, *} );
```
Die Tabelle laden, die von dem in der Variablen My_R_Script gespeicherten Skript zurückgegeben wird, wenn die Werte von B umbenannt in D, A und C gesendet werden. Durch Verwenden von * werden die restlichen nicht referenzierten Felder gesendet.

Bei der Dateierweiterung von DataFiles-Verbindungen wird zwischen Groß- und Kleinschreibung unterschieden. Beispiel: .qvd.

Formatoptionen

Die Formatbezeichnung besteht aus mehreren Formatoptionen, von denen jede eine bestimmte Eigenschaft der Tabelle definiert:

fspec-item ::= [**ansi | oem | mac | UTF-8 | Unicode | txt | fix | dif | biff | ooxml | html | xml | kml | qvd | qvx | parquet | delimiter is** `char` | **no eof | embedded labels | explicit labels | no labels | table is [**`tablename`**]** | **header is** `n` | **header is** `line` | **header is** `n` **lines** | **comment is** `string` | **record is** `n` | **record is** `line` | **record is** `n` **lines** | **no quotes |msq** | **URL is** `string` | **userAgent is** `string`]

Zeichensatz

Der Zeichensatz ist ein Dateispezifizierer für den **LOAD**-Befehl, mit dem der in der Datei verwendete Zeichensatz definiert wird.

Die Spezifizierer **ansi**, **oem** und **mac** wurden bereits in QlikView verwendet und funktionieren weiterhin. Sie werden jedoch nicht generiert, wenn der **LOAD**-Befehl mit Qlik Sense generiert wird.

Syntax:

```
utf8 | unicode | ansi | oem | mac | codepage is
```

Argumente:

Argumente

Argument	Beschreibung
utf8	UTF-8-Zeichensatz
unicode	Unicode-Zeichensatz
ansi	Windows, Codepage 1252
oem	DOS, OS/2, AS400 und andere
mac	Codepage 10000
codepage is	Mit dem Spezifizierer **codepage** kann jede Windows-Codepage als *N* verwendet werden.

Beschränkungen:

Die Konvertierung aus dem **oem**-Zeichensatz wird für macOS nicht implementiert. Ist kein Zeichensatz definiert, geht das Programm unter 1252 von Windows aus.

Beispiel:

```
LOAD * from a.txt (utf8, txt, delimiter is ',' , embedded labels)

LOAD * from a.txt (unicode, txt, delimiter is ',' , embedded labels)

LOAD * from a.txt (codepage is 10000, txt, delimiter is ',' , no labels)
```

Siehe auch:

Load (page 160)

Tabellenformat

Das Tabellenformat ist ein Dateispezifizierer für den **LOAD**-Befehl, der den Dateityp definiert. Ist kein Dateiformat angegeben, wird von einer *.txt*-Datei ausgegangen.

Tabellenformattypen

Typ	Beschreibung
txt	In einer Textdatei mit Trennzeichen werden die Spalten der Tabelle durch ein bestimmtes Trennzeichen voneinander getrennt.
fix	In einer Datei mit festen Datensatzlängen dagegen hat jedes Feld eine bestimmte Zahl von Zeichen. Dateien mit festen Datensatzlängen enthalten in der Regel mit einem Zeilenvorschub getrennte Datensätze. Es gibt aber auch erweiterte Optionen zur Angabe der Datensatzgröße in Byte oder die Möglichkeit, mit **Record is** mehrere Zeilen zu überspannen. *Enthalten die Daten Multi-Byte-Zeichen, kann die Ausrichtung von Feldumbrüchen fehlerhaft sein, weil das Format auf festen Längen in Byte basiert.*
dif	In *.dif*-Dateien (Data Interchange Format) liegt ein spezielles Tabellenformat vor.
biff	bQlik Sense kann auch Daten in Excel-Standarddateien durch das *biff*-Format (Binary Interchange File Format) interpretieren.
ooxml	Excel 2007 und spätere Versionen verwenden das Format ooxml *.xslx*.
html	Wenn die Tabelle Bestandteil einer html-Seite oder -Datei ist, muss html verwendet werden.
xml	Bei xml (Extensible Markup Language) handelt es sich um eine häufig verwendete Markup-Language, mit der Datenstrukturen im Textformat repräsentiert werden können.
qvd	Das Format *qvd* ist das systemeigene QVD-Dateiformat, das aus einer Qlik Sense App exportiert wird.
qvx	*qvx* ist ein Dateiformat für schnelles Einlesen von Daten in Qlik Sense.
parquet	Apache Parquetist ein Spalten-Speicherformat, das sehr effizient beim Speichern und Abfragen großer Datensätze ist.

Delimiter is

Für Tabellen mit Trennzeichen kann ein beliebiges Zeichen durch den Spezifizierer **delimiter is** definiert werden. Diese Formatoption ist nur für .txt-Dateien relevant.

Syntax:

```
delimiter is char
```

Argumente:

Argumente

Argument	Beschreibung
char	Steht für ein einzelnes Zeichen der 127 ASCII-Zeichen.

Darüber hinaus können folgende Werte verwendet werden:

Optionale Werte

Wert	Beschreibung
'\t'	mit oder ohne Anführungszeichen stellt einen Tabstopp dar.
'\\'	stellt einen umgekehrten Schrägstrich (\ ) dar.
'spaces'	stellt alle Kombinationen von einem oder mehreren Leerzeichen dar. Nicht druckbare Zeichen, deren ASCII-Wert niedriger als 32 ist, mit Ausnahme von CR und LF, werden als Leerzeichen interpretiert.

Fehlt diese Angabe, wird **delimiter is ','** angenommen.

Beispiel:

```
LOAD * from a.txt (utf8, txt, delimiter is ',' , embedded labels);
```

Siehe auch:

Load (page 160)

No eof

Die Formatoption **no eof** ignoriert das Zeichen für das Dateiende beim Laden von **.txt**-Dateien.

Syntax:

```
no eof
```

Wird der Spezifizierer **no eof** verwendet, werden Zeichen mit Codepoint 26, der häufig das Dateiende – End of File – markiert, ignoriert und können Teil eines Feldwerts sein.

Diese Option ist nur für Textdateien mit Trennzeichen relevant.

Beispiel:

```
LOAD * from a.txt (txt, utf8, embedded labels, delimiter is ' ', no eof);
```

Siehe auch:

Load (page 160)

Labels

Labels ist ein Dateispezifizierer für den **LOAD**-Befehl, mit dem die Stelle in einer Datei definiert wird, an der die Feldnamen gefunden werden können.

Syntax:

```
embedded labels|explicit labels|no labels
```

Die Feldnamen können an verschiedenen Stellen der Datei stehen. Enthält die erste Zeile der Tabelle die Feldnamen, benutzen Sie die Formatoption **embedded labels**. Wenn keine Feldnamen vorhanden sind, benutzen Sie **no labels**. In *dif*-Dateien sind die Feldnamen manchmal in einem separaten Tabellenkopf enthalten. In diesem Fall benutzen Sie die Formatoption **explicit labels**. Ist keine Formatoption bezüglich der Feldnamen angegeben, wird angenommen, dass diese in der ersten Zeile stehen, also **embedded labels**. Das gilt auch für *dif*-Dateien.

Example 1:

```
LOAD * from a.txt (unicode, txt, delimiter is ',' , embedded labels
```

Example 2:

```
LOAD * from a.txt (codePage is 1252, txt, delimiter is ',' , no labels)
```

Siehe auch:

Load (page 160)

Header is

Gibt die Größe des Headers in Tabellen an. Mit der Formatoption **header is** kann eine beliebige Headergröße definiert werden. Der Text des Headers wird von Qlik Sense nicht berücksichtigt.

Syntax:

```
header is n
```

```
header is line
```

```
header is n lines
```

Die Header-Länge kann in Byte (**header is n**) oder in Zeilen (**header is line** oder **header is n lines**) angegeben werden. **n** muss eine positive Ganzzahl sein, die die Header-Länge darstellt. Fehlt diese Angabe, wird **header is 0** angenommen. Die Formatoption **header is** ist nur für Tabellendateien von Bedeutung.

Beispiel:

Hierbei handelt es sich um das Beispiel einer Datenquellentabelle mit einer Header-Textzeile, die von Qlik Sense nicht als Daten interpretiert werden soll.

```
*Header line
Col1,Col2
a,B
c,D
```

Mithilfe des Spezifizierers **header is 1 lines** wird die erste Zeile nicht als Daten geladen. In diesem Beispiel weist der Spezifizierer **embedded labels** Qlik Sense dazu an, den Inhalt der ersten nicht ausgeschlossenen Zeile als Feldnamen zu betrachten.

```
LOAD Col1, Col2
FROM 'lib://files/header.txt'
(txt, embedded labels, delimiter is ',', msq, header is 1 lines);
```

Daraus ergibt sich eine Tabelle mit zwei Feldern, Col1 und Col2.

Siehe auch:

Load (page 160)

Record is

Für Dateien mit festen Satzlängen wird die Satzlänge durch die Formatoption **record is** festgelegt.

Syntax:

```
Record is n
Record is line
Record is n lines
```

Argumente:

Argumente

Argument	Beschreibung
n	Gibt die Satzlänge in Byte an.
line	Gibt die Datensatzlänge als eine Zeile an.
n lines	Gibt die Satzlänge in Zeilen an, wobei n eine positive ganze Zahl ist, welche die Satzlänge angibt.

Beschränkungen:

Die Formatoption **record is** ist nur für **fix**-Dateien von Bedeutung.

Siehe auch:

Load (page 160)

Quotes

Quotes sind Dateispezifizierer für den **LOAD**-Befehl, mit dem definiert wird, ob Anführungszeichen verwendet werden, und die Rangfolge zwischen Anführungszeichen und Trennzeichen festgelegt wird. Nur für Textdateien.

Syntax:

```
no quotes
```

```
msq
```

Wenn der Spezifizierer nicht angegeben wird, werden standardmäßige Anführungszeichen verwendet, d. h. es können die Anführungszeichen " " oder ' ' verwendet werden, allerdings nur, wenn diese Zeichen das erste und letzte nicht leere Zeichen eines Feldwerts bilden.

Argumente:

Argumente

Argument	Beschreibung
no quotes	Wird verwendet, wenn Anführungszeichen in Textdateien nicht zulässig sein sollen.
msq	Damit können Anführungszeichen in modernem Stil aktiviert werden, so dass in Feldern mehrzeiliger Inhalt ermöglicht wird. Felder mit Zeilenschaltung müssen zwischen doppelten Anführungszeichen stehen. Hinsichtlich der Option msq gilt die Einschränkung, dass einzelne doppelte Anführungszeichen ("), die als erstes oder letztes Zeichen im Feldinhalt angezeigt werden, als Beginn oder Ende des mehrzeiligen Inhalts interpretiert werden. Dies kann zu unberechenbaren Ergebnissen im geladenen Datensatz führen. In diesem Fall müssen Sie stattdessen standardmäßige Anführungszeichen verwenden und den Spezifizierer nicht angeben.

XML

Dieser Skriptspezifizierer wird beim Laden von xml-Dateien verwendet.Gültige Optionen für den **XML**-Spezifizierer sind in der Syntax aufgelistet.

DTD-Dateien können nicht in Qlik Sense geladen werden.

Syntax:

```
xmlsimple
```

Siehe auch:

Load (page 160)

KML

Dieser Skriptspezifizierer wird beim Laden von KML-Dateien für die Verwendung in einer Kartenvisualisierung eingesetzt.

Syntax:

```
kml
```

Die KML-Datei kann entweder durch Polygone dargestellte Gebietsdaten (z. B. Länder oder Regionen), Liniendaten (z. B. Wege oder Straßen) oder durch Punkte in der Form [geografische Länge, geografische Breite] wiedergegebene Punktdaten (z. B. Städte oder Orte) darstellen.

URL is

Dieser Skriptspezifizierer wird zum Festlegen der URL einer Webdatei-Datenverbindung beim Laden einer Webdatei verwendet.

Syntax:

```
URL is string
```

Argumente:

Argumente

Argument	Beschreibung
string	Gibt die URL der zu ladenden Datei an. Damit wird die URL überschrieben, die in der verwendeten Webdateiverbindung festgelegt ist.

Beschränkungen:

Die Formatoption **URL is** ist nur für Webdateien von Bedeutung. Sie müssen eine vorhandene Webdatei-Datenverbindung verwenden.

Siehe auch:

Load (page 160)

userAgent is

Dieser Skriptspezifikator wird verwendet, um den Browser-Benutzer-Agent beim Laden einer Webdatei festzulegen.

Syntax:

```
userAgent is string
```

Argumente:

Argumente

Argument	Beschreibung
string	Gibt den Benutzer-Agent-String des Browsers an. Dadurch wird der Standard-Benutzer-Agent des Browsers "Mozilla/5.0" überschrieben.

Beschränkungen:

Die Formatoption **userAgent is** ist nur für Webdateien von Bedeutung.

Siehe auch:

Load (page 160)

Let

Der **let**-Befehl ergänzt den **set**-Befehl und definiert die Skriptvariablen. Im Gegensatz zum **set**-Befehl wird beim **let**-Befehl der Ausdruck rechts des Gleichheitszeichens "=" zur Laufzeit des Skripts ausgewertet, bevor er der Variablen zugewiesen wird.

Syntax:

```
Let variablename=expression
```

Beispiele und Ergebnisse:

Beispiel	Ergebnis

`Set x=3+4;`
 	
`Let y=3+4;`
 	
`z=$(y)+1;`
	`$(x)` wird als ' 3+4 ' interpretiert
	
	`$(y)` wird als ' 7 ' interpretiert
	
	`$(z)` wird als ' 8 ' interpretiert
	
	Achten Sie auf den Unterschied zwischen den Befehlen **Set** und **Let**. Der Befehl **Set** weist den String „3+4“ zur Variablen zu, während der Befehl **Let** den String auswertet und der Variablen 7 zuweist.

Let T=now( );	`$(T)` liefert den Wert der aktuellen Uhrzeit.

Loosen Table

Durch den Befehl **Loosen Table** können eine oder mehrere interne Qlik Sense-Datentabellen während der Skriptausführung explizit als freie Tabellen eingestuft werden. Bei einer freien Tabelle werden alle Verknüpfungen zwischen Feldwerten in der Tabelle entfernt. Ein ähnlicher Effekt kann durch Laden jedes Felds der freien Tabellen als unabhängige, unverbundene Tabellen erzielt werden. Freie Tabellen können während des Testens nützlich sein, um verschiedene Teile der Datenstruktur vorübergehend zu isolieren. Im Tabellenmodell werden freie Tabellen durch gepunktete Linien dargestellt. Enthält das Skript eine oder mehrere **Loosen Table**-Befehle, werden zu Beginn der Skriptausführung alle vorher getroffenen Einstellungen bezüglich freier Tabellen in Qlik Sense aufgehoben.

Syntax:

```
Loosen Tabletablename [ , tablename2 ...]
```

```
Loosen Tablestablename [ , tablename2 ...]
```

Es kann entweder die Syntax **Loosen Table** oder **Loosen Tables** verwendet werden.

> *Hinweis: Entdeckt Qlik Sense bei der Ausführung des Skripts Zirkelbezüge in der Datenstruktur, die sich nicht durch Ihre Definition freier Tabellen auflösen, werden weitere Tabellen automatisch in freie Tabellen umgewandelt, solange bis kein Zirkelbezug mehr besteht. In diesem Fall werden Sie durch eine **Loop-Warnung** auf die Zirkelbezüge hingewiesen.*

Beispiel:

```
Tab1:

SELECT * from Trans;

Loosen Table Tab1;
```

Map

Die Funktion **map ... using** wird für die Zuordnung eines bestimmten Feldwerts oder einer Formel zu den Werten einer bestimmten Mapping-Tabelle verwendet. Die Mapping-Tabelle wird mit dem Befehl **Mapping** erstellt.

Syntax:

```
Map fieldlist Using  mapname
```

Die automatische Zuordnung wird für Felder ausgeführt, die nach dem Ende des Befehls **Map ... Using** bis zum Ende des Skripts geladen werden, oder bis der Befehl **Unmap** auftritt.

Die Zuordnung ist der letzte Schritt bei den Vorgängen, die zum Speichern des Felds in der internen Tabelle in Qlik Sense führen. Das bedeutet, dass das Zuordnen nicht jedes Mal geschieht, wenn ein Feldname als Teil einer Formel auftaucht, sondern erst direkt vor dem Speichern des Werts in einer programminternen Tabelle. Wird eine Zuordnung auf Formel-Ebene benötigt, muss die Funktion **Applymap()** verwendet werden.

Argumente:

Argumente

Argument	Beschreibung
fieldlist	Eine kommagetrennte Felderliste, der von dieser Stelle des Skripts an ein Tag zugewiesen werden sollte. Das Sternchen * steht für alle Felder. Die Wildcards * und ? sind in Feldnamen zugelassen. Beim Gebrauch von Wildcards innerhalb von Feldnamen müssen diese gegebenenfalls in Anführungszeichen stehen.
mapname	Der Name einer Mapping-Tabelle, die durch die Befehle **mapping load** oder **mapping select** geladen wurde.

Beispiele und Ergebnisse:

Beispiel	Ergebnis
Map Country Using Cmap;	Erlaubt das Zuordnen des Felds Country mit Cmap.
Map A, B, C Using X;	Erlaubt das Zuordnen der Felder A, B und C mit X.
Map * Using GenMap;	Erlaubt das Zuordnen aller Felder mit GenMap.

NullAsNull

Der Befehl **NullAsNull** schaltet die Konvertierung von NULL-Werten in Stringwerte aus, die zuvor mit dem Befehl **NullAsValue** festgelegt wurde.

Syntax:

```
NullAsNull *fieldlist
```

Der Befehl **NullAsValue** wird als Schalter angewendet und kann im Skript mehrmals mithilfe des Befehls **NullAsValue** oder **NullAsNull** ein- oder ausgeschaltet werden.

Argumente:

Argumente

Argument	Beschreibung
*fieldlist	Eine kommagetrennte Felderliste, für die **NullAsNull** aktiviert werden sollte. Das Sternchen * steht für alle Felder. Die Wildcards * und ? sind in Feldnamen zugelassen. Beim Gebrauch von Wildcards innerhalb von Feldnamen müssen diese gegebenenfalls in Anführungszeichen stehen.

Beispiel:

```
NullAsNull A,B;
LOAD A,B from x.csv;
```

NullAsValue

Mit dem Befehl **NullAsValue** werden die Felder angegeben, für die NULL-Werte in Werte konvertiert werden sollen.

Syntax:

```
NullAsValue *fieldlist
```

Qlik Sense behandelt NULL-Werte standardmäßig als fehlende oder nicht definierte Werte. Je nach Kontext sind NULL-Werte aber unter Umständen als besondere Werte anstatt einfach als fehlende Werte zu interpretieren. So kann man mit dem NULL-Befehl etwa die Verknüpfung zwischen NULL-Werten und anderen **NullAsValue**-Werten, die normalerweise nicht möglich ist, zulassen.

Der Befehl **NullAsValue** wirkt als Schalter und wird an den folgenden Load-Befehlen ausgeführt. Er kann über den **NullAsNull**-Befehl wieder deaktiviert werden.

Argumente:

Argumente

Argument	Beschreibung
*fieldlist	Eine kommagetrennte Felderliste, für die **NullAsValue** aktiviert werden sollte. Das Sternchen * steht für alle Felder. Die Wildcards * und ? sind in Feldnamen zugelassen. Beim Gebrauch von Wildcards innerhalb von Feldnamen müssen diese gegebenenfalls in Anführungszeichen stehen.

Beispiel:

```
NullAsValue A,B;
Set NullValue = 'NULL';
LOAD A,B from x.csv;
```

Qualify

Mit dem Befehl **Qualify** wird die Qualifikation von Feldnamen aktiviert, d. h. Feldnamen erhalten den Tabellennamen als Zusatz.

Syntax:

```
Qualify *fieldlist
```

Die automatische Verknüpfung von gleichnamigen Feldern in unterschiedlichen Tabellen kann mit dem Befehl **qualify** verhindert werden, indem die Feldnamen durch den Tabellennamen ergänzt und somit qualifiziert werden. Die Umbenennung erfolgt, wenn der Feldname in der Tabelle gefunden wird. Der neue Name hat das Format *tablename.fieldname*. *Tablename* entspricht der Bezeichnung der aktuellen Tabelle, bzw. falls keine Bezeichnung vorhanden ist, dem Namen, der nach **from** in **LOAD**- und **SELECT**-Befehlen angezeigt wird.

Die Qualifizierung erfolgt für alle Felder, die nach dem Befehl **qualify** geladen wurden.

Die Qualifizierung ist standardmäßig zu Beginn der Skriptausführung immer deaktiviert. Die Qualifizierung eines Feldnamens kann jederzeit anhand eines **qualify**-Befehls erfolgen. Mit einem **Unqualify**-Befehl kann die Qualifizierung jederzeit deaktiviert werden.

*Hinweis: Der **qualify**-Befehl sollte nicht in Zusammenhang mit einer partiellen Ausführung des Skripts verwendet werden.*

Argumente:

Argumente

Argument	Beschreibung
*fieldlist	Eine kommagetrennte Felderliste, für welche die Qualifizierung aktiviert werden sollte. Das Sternchen * steht für alle Felder. Die Wildcards * und ? sind in Feldnamen zugelassen. Beim Gebrauch von Wildcards innerhalb von Feldnamen müssen diese gegebenenfalls in Anführungszeichen stehen.

Example 1:

```
Qualify B;

LOAD A,B from x.csv;

LOAD A,B from y.csv;
```

Die beiden Tabellen **x.csv** und **y.csv** sind nur durch **A** verknüpft. Daraus resultieren drei Felder: A, x.B, y.B.

Example 2:

Insbesondere in Datenbanken, deren Struktur nicht bekannt ist, möchten Sie möglicherweise nur Verknüpfungen über ein einziges oder wenige Felder zulassen. Dieser Vorgang wird im folgenden Beispiel dargestellt:

```
qualify *;

unqualify TransID;

SQL SELECT * from tab1;

SQL SELECT * from tab2;

SQL SELECT * from tab3;
```

In diesem Fall werden die Tabellen **TransID**, *tab1* und *tab2* nur über das Feld *tab3* verknüpft.

Rem

Der **rem**-Befehl dient dazu, Anmerkungen oder Kommentare in das Skript einzufügen oder Befehle vorübergehend zu deaktivieren, ohne sie aus dem Skript zu löschen.

Syntax:

```
Rem string
```

Sämtliche Zeichen zwischen **rem** und dem nächsten Semikolon **;** gelten als Kommentar.

Es gibt zwei weitere Möglichkeiten, Kommentare im Skript einzufügen:

1. Sie können an beliebiger Stelle im Skript – nicht jedoch zwischen zwei Anführungszeichen – Kommentare einfügen, indem Sie sie zwischen **/*** und ***/** einschließen.
2. Wenn Sie **//** im Skript eingeben, wird der Rest der Zeile als Kommentar betrachtet. (Mit der Ausnahme, dass //: auch in Internet-Adressen verwendet werden kann.)

Argumente:

Argumente

Argument	Beschreibung
string	Beliebiger Text.

Beispiel:

```
Rem ** This is a comment **;
/* This is also a comment */
// This is a comment as well
```

Rename

Das Skriptschlüsselwort **Rename** kann zum Umbenennen von bereits geladenen Tabellen oder Feldern verwendet werden.

Rename field

Diese Skriptfunktion benennt nach dem Laden ein oder mehrere bestehende Qlik Sense-Felder um.

> *Es wird nicht empfohlen, für eine Variable in Qlik Sense denselben Namen wie für ein Feld oder eine Funktion zu verwenden.*

Es kann entweder die Syntax **rename field** oder **rename fields** verwendet werden.

Syntax:

```
Rename Field (using mapname | oldname to newname{ , oldname to newname })
```

```
Rename Fields (using mapname | oldname to newname{ , oldname to newname })
```

Argumente:

Argument	Beschreibung
mapname	Der Name einer bereits geladenen Mapping-Tabelle, die eine Spalte mit alten und eine mit neuen Feldnamen enthält.
oldname	Der alte Feldname.
newname	Der neue Feldname.

Beschränkungen:

Sie können zwei Felder nicht auf denselben Namen umbenennen.

Example 1:

```
Rename Field XAZ0007 to Sales;
```

Example 2:

```
FieldMap:

Mapping SQL SELECT oldnames, newnames from datadictionary;

Rename Fields using FieldMap;
```

Rename table

Diese Skriptfunktion benennt nach dem Laden eine oder mehrere bestehende interne Qlik Sense-Tabellen um.

Es kann entweder die Syntax **rename table** oder **rename tables** verwendet werden.

Syntax:

```
Rename Table (using mapname | oldname to newname{ , oldname to newname })
Rename Tables (using mapname | oldname to newname{ , oldname to newname })
```

Argumente:

Argumente

Argument	Beschreibung
mapname	Der Name einer bereits geladenen Mapping-Tabelle, die eine Spalte mit alten und eine mit neuen Tabellennamen enthält.
oldname	Der alte Tabellenname.
newname	Der neue Tabellenname.

Beschränkungen:

Zwei Tabellen mit unterschiedlichen Namen können nach dem Umbenennen nicht den gleichen Namen haben. Das Skript erzeugt einen Fehler, wenn Sie versuchen, die Tabelle umzubenennen und bereits eine Tabelle mit demselben Namen vorhanden ist.

Example 1:

```
Tab1:
SELECT * from Trans;
Rename Table Tab1 to Xyz;
```

Example 2:

```
TabMap:
Mapping LOAD oldnames, newnames from tabnames.csv;
Rename Tables using TabMap;
```

Search

Der **Search**-Befehl wird zum Ein- und Ausschließen von Feldern bei der intelligenten Suche verwendet.

Syntax:

```
Search Include *fieldlist
Search Exclude *fieldlist
```

Sie können mehrere Search-Befehle verwenden, um die Auswahl der Felder, die in der Suche enthalten sein sollen, zu verfeinern. Die Befehle werden von oben nach unten ausgewertet.

Argumente:

Argumente

Argument	Beschreibung
*fieldlist	Eine kommagetrennte Liste der Felder, die in der intelligenten Suche eingeschlossen oder von ihr ausgeschlossen werden sollen. Das Sternchen * steht für alle Felder. Die Wildcards * und ? sind in Feldnamen zugelassen. Beim Gebrauch von Wildcards innerhalb von Feldnamen müssen diese gegebenenfalls in Anführungszeichen stehen.

Beispiel:

Beispiele für Suchen

Befehl	Beschreibung
`Search Include *;`	Alle Felder bei der intelligenten Suche berücksichtigen.
`Search Exclude [*ID];`	Alle Felder, die mit ID enden, aus der intelligenten Suche ausschließen.
`Search Exclude '*ID';`	Alle Felder, die mit ID enden, aus der intelligenten Suche ausschließen.
`Search Include ProductID;`	Das Feld ProductID bei der intelligenten Suche berücksichtigen.

Das kombinierte Resultat dieser drei Befehle in dieser Reihenfolge ist, dass alle mit ID endenden Felder, ausgenommen ProductID, aus der intelligenten Suche ausgeschlossen werden.

Section

Mit dem **section**-Befehl können Sie definieren, ob die nachfolgenden Befehle **LOAD** und **SELECT** Daten laden oder aber eine Zugriffskontrollen festlegen.

Syntax:

```
Section (access | application)
```

Fehlt diese Angabe, wird **section application** angenommen. Die **section**-Definition bleibt solange gültig, bis ein neuer **section**-Befehl erscheint.

Beispiel:

```
Section access;

Section application;
```

Select

Die Auswahl von Feldern aus einer ODBC-Datenquelle oder aus einem OLE DB-Provider erfolgt über die gewöhnlichen SQL-**SELECT**-Befehle. Allerdings hängt die Akzeptanz von **SELECT**-Befehlen vom verwendeten ODBC-Treiber oder OLE DB-Provider ab. Für die Verwendung der **SELECT**-Anweisung ist eine offene Datenverbindung zur Quelle erforderlich.

Syntax:

```
Select [all | distinct | distinctrow | top n [percent] ] fieldlist

From tablelist

[where criterion ]

[group by fieldlist [having criterion ] ]

[order by fieldlist [asc | desc] ]

[ (Inner | Left | Right | Full) join tablename on fieldref = fieldref ]
```

Ferner können mehrere **SELECT**-Anweisungen manchmal zu einer zusammengefasst werden. Dies geschieht durch den Operator **union**:

```
selectstatement Union selectstatement
```

Der **SELECT**-Befehl wird vom ODBC-Treiber oder OLE DB-Provider interpretiert, deshalb kann es je nach Leistungsfähigkeit des SQL-Treibers oder ODBC-Providers gelegentlich Abweichungen von der allgemeinen OLE DB-Syntax geben:

- **as** wird manchmal nicht akzeptiert, d. h. *aliasname* muss direkt nach *fieldname* stehen.
- **as** ist manchmal zwingend notwendig, wenn ein *aliasname* verwendet wird.
- **distinct**, **as**,**where**, **group by**, **order by** oder **union** wird manchmal nicht unterstützt.
- Einige ODBC-Treiber akzeptieren nicht alle oben genannten Anführungszeichen.

*Der **SELECT**-Befehl in SQL kann hier nicht bis in alle Einzelheiten beschrieben werden. Beispielsweise können **SELECT**-Befehle verschachtelt werden; es können innerhalb eines **SELECT**-Befehls Verknüpfungen vorgenommen werden; es gibt noch weitere Funktionen, die in Formeln verwendet werden können, usw.*

Argumente:

Argumente

Argument	Beschreibung
distinct	**distinct** ist ein Prädikat, das verwendet wird, wenn doppelte Kombinationen von Werten in den ausgewählten Feldern nur einmal geladen werden sollen.
distinctrow	**distinctrow** ist ein Prädikat, mit dem doppelte Datensätze in der Quelltabelle nur einmal geladen werden.
fieldlist	**fieldlist ::= (*| field) {, field }** Liste der zu ladenden Felder. * in der Felderliste bedeutet alle Felder in der Tabelle. **fieldlist ::= field {, field }** Eine Auflistung eines oder mehrerer Felder, durch Kommas voneinander getrennt. **field ::= (fieldref| expression) [as aliasname]** Die Formel kann eine numerische Funktion oder Stringfunktion sein, die sich auf ein oder mehrere andere Felder bezieht. Dies sind einige der Operatoren und Funktionen, die in der Regel akzeptiert werden: +, -, *, /, & (String-Zusammenfassung), sum(fieldname), count (fieldname), avg(fieldname)(average), month(fieldname) usw. Weitere Informationen finden Sie in der Dokumentation des ODBC-Treibers. **fieldref ::= [tablename.] fieldname** Der **tablename** und der **fieldname** sind Textstrings, die ihrem Namen entsprechen. Sie müssen von geraden doppelten Anführungszeichen eingeschlossen sein, wenn sie z. B. Leerzeichen enthalten. Die Bedingung **as** weist dem Feld einen neuen Namen zu.
from	**tablelist ::= table {, table }** Eine Liste der Tabellen, aus denen Felder geladen werden sollen. **table ::= tablename [[as] aliasname]** Der **tablename** kann, muss aber nicht in Anführungszeichen stehen.
where	**where** wird benutzt, um anhand eines Kriteriums zu prüfen, ob der Datensatz geladen wird oder nicht. **criterion** ist eine logische Formel, die mitunter sehr komplex sein kann. Folgende Operatoren sind normalerweise möglich: numerische Operatoren und Funktionen, **=**, **<>** oder **#(**ungleich), **>**, **>=**, **<**, **<=**, **and**, **or**,**not**, **exists**,**some**, **all**,**in** oder auch neue **SELECT**-Anweisungen. Weitere Informationen finden Sie in der Dokumentation des ODBC-Treibers oder OLE DB-Providers.
group by	Der Zusatz **group by** dient dazu, Datensätze zu aggregieren (gruppieren). Innerhalb einer Gruppe müssen alle Datensätze in einem bestimmten Feld denselben Wert haben. Anderenfalls kann das Feld lediglich innerhalb von Formeln, z. B. als Summe oder Durchschnitt, benutzt werden.
having	Der Zusatz **having** hat für Gruppen dieselbe Bedeutung wie der Zusatz **where** für Datensätze.

Argument	Beschreibung
order by	Mit **order by** können Sie eine Sortierfolge für die durch den **SELECT**-Befehl entstehende Tabelle festlegen.
join	**join** wird benutzt, um mehrere Tabellen zu einer zusammenzuschließen. Feld- und Tabellennamen müssen in Anführungszeichen stehen, wenn sie Leerzeichen oder Buchstaben aus nationalen Zeichensätzen enthalten. Wenn das Skript automatisch von Qlik Sense erstellt wird, werden stets Anführungszeichen verwendet, die auf den ODBC-Treiber oder den OLE DB-Provider abgestimmt sind, der in der Datenquellen-Definition der Datenquelle im Befehl **Connect** angegeben ist.

Example 1:

```
SELECT * FROM `Categories`;
```

Example 2:

```
SELECT `Category ID`, `Category Name` FROM `Categories`;
```

Example 3:

```
SELECT `Order ID`, `Product ID`,
`Unit Price` * Quantity * (1-Discount) as NetSales
FROM `Order Details`;
```

Example 4:

```
SELECT `Order Details`.`Order ID`,
Sum(`Order Details`.`Unit Price` * `Order Details`.Quantity) as `Result`
FROM `Order Details`, Orders
where Orders.`Order ID` = `Order Details`.`Order ID`
group by `Order Details`.`Order ID`;
```

Set

Der Befehl **set** wird zum Festlegen der Skriptvariablen verwendet. Diese können Strings, Pfade, Laufwerke usw. im Skript ersetzen.

Syntax:

```
Set variablename=string
```

Example 1:

```
Set FileToUse=Data1.csv;
```

Example 2:

```
Set Constant="My string";
```

Example 3:

```
Set BudgetYear=2012;
```

Sleep

Der Befehl **sleep** hält die Ausführung des Skripts für eine festgelegte Zeit an.

Syntax:

```
Sleep n
```

Argumente:

Argument	Beschreibung
n	In Millisekunden angegeben, wobei *n* für eine positive ganze Zahl steht, die nicht größer als 3600000 sein darf (d. h. 1 Stunde). n kann auch durch eine Formel berechnet werden.

Example 1:

```
sleep 10000;
```

Example 2:

```
sleep t*1000;
```

SQL

Mithilfe der **SQL**-Anweisung können Sie einen beliebigen SQL-Befehl über eine ODBC- oder OLE DB-Verbindung senden.

Syntax:

```
SQL sql_command
```

Beim Senden von SQL-Befehlen zur Aktualisierung der Datenbank wird eine Fehlermeldung zurückgegeben, wenn Qlik Sense die ODBC-Verbindung im schreibgeschützten Modus geöffnet hat.

Die Syntax:

```
SQL SELECT * from tab1;
```

ist zulässig und ersetzt aus Konsistenzgründen den herkömmlichen **SELECT**-Befehl. Der Zusatz SQL vor **SELECT**-Befehlen bleibt jedoch optional.

Argumente:

Argument	Beschreibung
sql_command	Ein gültiger SQL-Befehl.

Example 1:

```
SQL leave;
```

Example 2:

```
SQL Execute <storedProc>;
```

SQLColumns

Der Befehl **sqlcolumns** erzeugt eine Reihe von Feldern, welche die Spalten der ODBC- oder OLE DB-Datenquelle beschreiben, zu der mit dem **connect**-Befehl eine Verbindung hergestellt wurde.

Syntax:

```
SQLcolumns
```

Die Felder geben, zusammen mit den durch die Befehle **sqltables** und **sqltypes** generierten Feldern, genauen Aufschluss über eine angegebene Datenbank. Die zwölf Standardfelder lauten:

TABLE_QUALIFIER

TABLE_OWNER

TABLE_NAME

COLUMN_NAME

DATA_TYPE

TYPE_NAME

PRECISION

LENGTH

SCALE

RADIX

NULLABLE

REMARKS

Eine detaillierte Beschreibung dieser Felder finden Sie in Ihrer ODBC-Dokumentation.

Beispiel:

```
Connect to 'MS Access 7.0 Database; DBQ=C:\Course3\DataSrc\QWT.mbd';
SQLcolumns;
```

Manche ODBC-Treiber unterstützen diesen Befehl eventuell nicht. Manche ODBC-Treiber legen unter Umständen weitere Felder an.

SQLTables

Der Befehl **sqltables** erzeugt eine Reihe von Feldern, welche die Tabellen einer ODBC- oder OLE DB-Datenquelle beschreiben, zu der mit dem **connect**-Befehl eine Verbindung hergestellt wurde.

Syntax:

```
SQLTables
```

Die Felder geben, zusammen mit den durch die Befehle **sqlcolumns** und **sqltypes** generierten Feldern, genauen Aufschluss über eine angegebene Datenbank. Die fünf Standardfelder lauten:

TABLE_QUALIFIER

TABLE_OWNER

TABLE_NAME

TABLE_TYPE

REMARKS

Eine detaillierte Beschreibung dieser Felder finden Sie in Ihrer ODBC-Dokumentation.

Beispiel:

```
Connect to 'MS Access 7.0 Database; DBQ=C:\Course3\DataSrc\QWT.mbd';
SQLTables;
```

Manche ODBC-Treiber unterstützen diesen Befehl eventuell nicht. Manche ODBC-Treiber legen unter Umständen weitere Felder an.

SQLTypes

Der Befehl **sqltypes** erzeugt eine Reihe von Feldern, welche die Arten der ODBC- oder OLE DB-Datenquelle beschreiben, zu der mit dem **connect**-Befehl eine Verbindung hergestellt wurde.

Syntax:

```
SQLTypes
```

Die Felder geben, zusammen mit den durch die Befehle **sqlcolumns** und **sqltables** generierten Feldern, genauen Aufschluss über eine angegebene Datenbank. Die fünfzehn Standardfelder lauten:

TYPE_NAME

DATA_TYPE

PRECISION

LITERAL_PREFIX

LITERAL_SUFFIX

CREATE_PARAMS

NULLABLE

CASE_SENSITIVE

SEARCHABLE

UNSIGNED_ATTRIBUTE

MONEY

AUTO_INCREMENT

LOCAL_TYPE_NAME

MINIMUM_SCALE

MAXIMUM_SCALE

Eine detaillierte Beschreibung dieser Felder finden Sie in Ihrer ODBC-Dokumentation.

Beispiel:

```
Connect to 'MS Access 7.0 Database; DBQ=C:\Course3\DataSrc\QWT.mbd';
SQLTypes;
```

Manche ODBC-Treiber unterstützen diesen Befehl eventuell nicht. Manche ODBC-Treiber legen unter Umständen weitere Felder an.

Star

Der String, der alle Werte eines Feldes repräsentieren soll (Stern-Symbol), kann mit dem **star**-Befehl definiert werden. Dies hat Gültigkeit für alle nachfolgenden **LOAD**- und **SELECT**-Befehle.

Syntax:

```
Star is[ string ]
```

Argumente:

Argumente

Argument	Beschreibung
string	Beliebiger Text. Beachten Sie, dass der String in Anführungszeichen stehen muss, wenn er Leerzeichen enthält. Wird kein solcher Befehl verwendet, wird **star is;** angenommen, d. h. es ist kein Stern-Symbol verfügbar, solange es nicht ausdrücklich definiert wird. Das angegebene Stern-Symbol bleibt gültig, bis es durch einen neuen **star**-Befehl geändert wird.

Der **Star is**-Befehl sollte nicht im Datenteil des Skripts (unter **Abschnittsanwendung**) genutzt werden, wenn Abschnittszugriff verwendet wird. Für die geschützten Felder im **Section Access**-Teil des Skripts wird das Stern-Zeichen aber vollständig unterstützt. In diesem Fall benötigen Sie den expliziten **Star is**-Befehl nicht, da er in Section Access immer implizit enthalten ist.

Beschränkungen

- In Schlüsselfeldern kann das Stern-Zeichen nicht verwendet werden, d. h. in Feldern, die Tabellen verknüpfen.
- Das Stern-Zeichen kann in keinen Feldern verwendet sind, die vom **Unqualify**-Befehl betroffen sind, da sich dies auf Felder auswirken kann, die Tabellen verknüpfen.
- Das Stern-Zeichen kann nicht in nichtlogischen Tabellen verwendet werden, beispielsweise info-load-Tabellen oder mapping-load-Tabellen.
- Wenn das Stern-Zeichen in einem reduzierenden Feld (einem Feld, das mit den Daten verknüpft) in Section Access verwendet wird, repräsentiert es die Werte, die in diesem Feld in Section Access aufgelistet sind. Es repräsentiert keine anderen Werte, die in den Daten vorliegen könnten, aber nicht in Section Access aufgelistet sind.
- Sie können das Stern-Zeichen nicht in Feldern verwenden, die von irgend einer Form von Datenreduzierung außerhalb des **Section Access**-Bereichs betroffen sind.

Beispiel

Das nachfolgende Beispiel ist ein Ausschnitt eines Datenladeskripts mit Abschnittszugriff.

```
Star is *;



Section Access;

LOAD * INLINE [

ACCESS, USERID, OMIT

ADMIN, ADMIN,

USER, USER1, SALES

USER, USER2, WAREHOUSE

USER, USER3, EMPLOYEES

USER, USER4, SALES

USER, USER4, WAREHOUSE

USER, USER5, *

];
```

```
Section Application;

LOAD * INLINE [

SALES, WAREHOUSE, EMPLOYEES, ORDERS

1, 2, 3, 4

];
```

Dabei gilt Folgendes:

- Das Zeichen *Star* ist *.
- Dem Benutzer *ADMIN* werden alle Felder angezeigt. Es wird nichts ausgelassen.
- Der Benutzer *USER1* kann das Feld *SALES* nicht sehen.
- Der Benutzer *USER2* kann das Feld *WAREHOUSE* nicht sehen.
- Der Benutzer *USER3* kann das Feld *EMPLOYEES* nicht sehen.
- Der Benutzer *USER4* wird der Lösung zweimal hinzugefügt, damit zwei Felder für diesen Benutzer ausgelassen (OMIT)werden: *SALES* und *WAREHOUSE*.
- Dem Benutzer *USER5* wird ein „*“ hinzugefügt, was bedeutet, dass alle in OMIT aufgelisteten Felder nicht verfügbar sind, d. h., der Benutzer *USER5* kann die Felder *SALES*, *WAREHOUSE* und *EMPLOYEES* nicht sehen, aber der Benutzer kann das Feld *ORDERS* sehen.

Store

Mit dem Befehl **Store** wird eine QVD-, Parquet-, CSV- oder TXT-Datei erstellt.

Syntax:

```
Store [ fieldlist from] table into filename [ format-spec ];
```

Mit dem Befehl wird eine explizit benannte QVD-Datei, eine Parquet-Datei oder eine Textdatei erstellt.

Durch den Befehl werden Werte einer Datentabelle in die neue Datei exportiert. Wenn Felder aus mehreren Tabellen exportiert werden sollen, muss zuvor im Skript ein expliziter join erstellt werden, um die zu exportierende Datentabelle zu generieren.

Textwerte werden im CSV-Format nach UTF-8 exportiert. Es kann ein Trennzeichen festgelegt werden (siehe **LOAD**). Der **store**-Befehl in einer CSV-Datei unterstützt keinen BIFF-Export.

In einigen Fällen werden Felder, wenn Daten nicht korrekt formatiert sind, von doppelten Anführungszeichen eingeschlossen, um sicherzustellen, dass die Daten richtig interpretiert werden. Das kann beispielsweise vorkommen, wenn ein Feld Zeichen wie Anführungszeichen, Kommas, Leerzeichen oder Zeilenumbrüche enthält.

Argumente:

Argumente des Store-Befehls

Argument	Beschreibung
*fieldlist::= (* | field) { , field })*	Liste der zu ladenden Felder. Das Sternchen * steht für alle Felder. *field::= fieldname [***as** *aliasname]* Dabei ist *fieldname* ein Text, der einem Feldnamen in *table* entspricht. (Beachten Sie, dass der Feldname zwischen geraden doppelten Anführungszeichen oder eckigen Klammern stehen muss, wenn er Leerzeichen oder andere nicht standardmäßige Zeichen enthält.) *aliasname* ist ein alternativer Name, der anstelle des Feldnamens in der QVD- oder CSV-Datei verwendet werden soll.
table	Ein Skriptname einer bereits eingelesenen Tabelle.
filename	Der Name der Zieldatei einschließlich eines gültigen Pfads zu einer vorhandenen Ordner-Datenverbindung. **Beispiel: *'lib://Table Files/target.qvd'*** Im Legacymodus für die Skripterstellung werden die folgenden Pfadformate ebenfalls unterstützt: • absolut **Beispiel: *c:\data\sales.qvd*** • relativ zum Qlik Sense App-Arbeitsverzeichnis. **Beispiel: *data\sales.qvd*** Ist kein Pfad angegeben, speichert Qlik Sense die Datei in dem Verzeichnis, das in der **Directory**-Anweisung angegeben ist. Wenn keine **Directory**-Anweisung vorhanden ist, sucht Qlik Sense im Arbeitsverzeichnis *C:\Users\{user}\Documents\Qlik\Sense\Apps*. •

Argument	Beschreibung
*format-spec ::=***(** (**txt** | **qvd** | **parquet**), **Komprimierung ist** *codec***)**	Sie können die Formatspezifikation auf jedes dieser Dateiformate festlegen. Fehlt die Formatbezeichnung, wird **qvd** angenommen. • **txt** für CSV- und TXT-Dateien. • **qvd** für QVD-Dateien. • **parquet** für Parquet-Dateien. Wenn Sie **parquet** verwenden, können Sie auch festlegen, welcher Komprimierungs-Codec mit **Komprimierung ist** verwendet wird. Wenn Sie den Komprimierungs-Codec nicht mit **Komprimierung ist** festlegen, wird `snappy` verwendet. Die folgenden Komprimierungseinstellungen sind verfügbar: • `uncompressed` • `snappy` • `gzip` • `lz4` • `brotli` • `zstd` • `lz4_hadoop` Beispiel: `Store mytable into [lib://DataFiles/myfile.parquet] (parquet, compression is lz4);`

Beispiele:

```
Store mytable into xyz.qvd (qvd);

Store * from mytable into 'lib://FolderConnection/myfile.qvd';

Store Name, RegNo from mytable into xyz.qvd;

Store Name as a, RegNo as b from mytable into 'lib://FolderConnection/myfile.qvd';

Store mytable into myfile.txt (txt);

Store mytable into myfile.parquet (parquet);

Store * from mytable into 'lib://FolderConnection/myfile.qvd';
```

Bei der Dateierweiterung von DataFiles-Verbindungen wird zwischen Groß- und Kleinschreibung unterschieden. Beispiel: .qvd.

Speichern in Parquet-Dateien

Parquet ist ein stark typorientiertes Dateiformat, in dem jedes Feld einen einzelnen spezifischen Datentyp enthält (z. B. in32, double, timestamp oder text). Qlik Sense speichert interne Daten dual lose nach Typen sortiert, wobei Daten aus verschiedenen Quellen in den gleichen Feldern gemischt werden können. Da nur ein Teil des dualen Werts in jedem Feld in Parquet gespeichert werden kann, ist es wichtig zu wissen, was in jedem Feld enthalten ist. Standardmäßig verwendet Qlik Sense den Feldtyp, um zu bestimmen, wie das Feld gespeichert werden soll. Wenn Daten in Parquet-Dateien in einem spezifischen Format gespeichert werden, müssen, Sie beim Laden angeben, von welchem Datentyp Ihre Felder sind. Wenn Sie versuchen, Daten in einer Parquet-Datei in inkompatiblen Felder zu speichern (z. B. Zahlen in einem Textfeld oder Text in einem Zeitstempelfeld), führt dies zu Null-Werten.

Wenn Sie Daten laden, die in Parquet gespeichert werden sollen, ist es möglich, das Standardverhalten zu ändern. Sie können die Daten entweder formatieren, um den Datentyp zu ändern, oder sie mit Tags versehen, um einen bestimmten Spaltentyp in Parquet zu erzwingen.

Formatieren von Daten zum Speichern in Parquet

Sie können die Formatierungsfunktionen von Qlik Sense verwenden, um Ihre Daten zu klassifizieren. Beispielsweise können **Text()**, **Num()**, **Interval()** oder **Timestamp()** Datenformate beim Speichern von Daten in Parquet erzwingen. Qlik Sense kann Daten in fast 20 Datentyen speichern, abhängig von Feldattributen und automatischen Feld-Tags. Weitere Informationen finden Sie unter *Interpretationsfunktionen (page 1291)*.

Beispiel: Formatieren von Daten mit Num() und Text()

Das folgende Beispiel zeigt, wie Daten für das Speichern in Parquet vorbereitet werden. **Num()** wird auf das numerische Feld angewendet. **Text()** wird sowohl auf Textfelder als auch auf gemischte Felder angewendet. Im Fall gemischter Felder verhindert **Text()**, dass die Felder in Parquet wie numerische Felder behandelt und die Textwerte in Null-Werte geändert werden.

```
Data:
LOAD * INLINE [
num, text, mixed
123.321, abc, 123
456.654, def, xyz
789.987, ghi, 321
];

Format:
NoConcatenate
LOAD num, text, Text(mixed) as mixed RESIDENT Data;
STORE Format INTO [lib://DataFiles/Tmp.parquet] (parquet);
```

Taggen von Daten zum Speichern in Parquet

Sie taggen Ihre Daten mit `$parquet`-Tags, um spezifische Spaltentypen zu erzwingen, wenn Daten in Parquet gespeichert werden. Jeder Datentyp kann erzwungen werden, indem das entsprechende Kontroll-Tag hinzugefügt wird. Um beispielsweise ein Feld in Parquet als INT32 zu speichern, taggen Sie es im Ladeskript mit `$parquet-int32`. Abhängig vom Datentyp wird entweder die Zeichenfolge oder die numerische Darstellung der dualen Daten gespeichert.

Die folgenden Parquet-Kontroll-Tags können verwendet werden, um Felder zum Speichern in Parquet-Dateien zu taggen.

Parquet-Kontroll-Tags

Kontroll-Tag	Dual	Physischer Typ	Logischer Typ	Konvertierter Typ
$parquet-boolean	Zahl	BOOLEAN	NONE	NONE
$parquet-int32	Zahl	INT32	NONE	NONE
$parquet-int64	Zahl	INT64	NONE	NONE
$parquet-float	Zahl	FLOAT	NONE	NONE
$parquet-double	Zahl	DOUBLE	NONE	NONE
$parquet-bytearray	String	BYTE_ARRAY	NONE	UTF8
$parquet-bytearrayfix	Zahl	FIXED_LEN_BYTE_ARRAY	NONE	DECIMAL
$parquet-decimal	Zahl	INT64	DECIMAL	DECIMAL
$parquet-date	Zahl	INT32	DATE	DATE
$parquet-time	Zahl	INT64	TIME	TIME_MICROS
$parquet-timestamp	Zahl	INT64	TIMESTAMP	TIMESTAMP_MICROS
$parquet-string	String	BYTE_ARRAY	STRING	UTF8
$parquet-enum	String	BYTE_ARRAY	ENUM	ENUM
$parquet-interval	Zahl	FIXED_LEN_BYTE_ARRAY	INTERVAL	INTERVAL
$parquet-json	String	BYTE_ARRAY	JSON	JSON
$parquet-bson	String	BYTE_ARRAY	BSON	BSON
$parquet-uuid	String	FIXED_LEN_BYTE_ARRAY	UUID	NONE

Beispiel: Taggen von Daten zum Speichern in Parquet

In diesem Beispiel werden zwei Tags verwendet, um die Daten für Parquet zu definieren. Das Feld *num* wird mit `$parquet-int32` getaggt, um es als numerisches Feld zu definieren, das in Parquet als INT32 festgelegt wird.

```
Data:
LOAD * INLINE [
num, text,
123.321, abc
456.654, def
789.987, ghi
];
TAG num WITH '$parquet-int32';
STORE Data INTO [lib://DataFiles/Tmp.parquet] (parquet);
```

Table/Tables

Die Skriptschlüsselwörter **Table** und **Tables** werden in den Befehlen **Drop**, **Comment** und **Rename** sowie als Formatspezifizierer in **Load**-Befehlen verwendet.

Tag

Dieser Skriptbefehl bietet eine Möglichkeit zum Zuweisen von Tags zu einem oder mehreren Feldern oder Tabellen. Wird der Versuch unternommen, einem Feld oder einer Tabelle ein Tag hinzuzufügen, das bzw. die nicht in der App vorhanden ist, wird dieser Vorgang ignoriert. Gibt es Konflikte durch mehrfach vorkommende Feldnamen oder Tags, wird der letzte Tag verwendet.

Syntax:

```
Tag [field|fields] fieldlist with tagname
```

```
Tag [field|fields] fieldlist using mapname
```

```
Tag table tablelist with tagname
```

Argumente

Argument	Beschreibung
fieldlist	Ein oder mehrere Felder, denen ein Tag zugewiesen sein sollte, in einer kommagetrennten Liste.
mapname	Der Name einer Mapping-Tabelle, die durch die Befehle **mapping Load** oder **mapping Select** geladen wurde.
tablelist	Eine kommagetrennte Liste der Tabellen, denen ein Tag zugewiesen werden sollte.
tagname	Der Name des Tags, das auf das Feld angewendet werden sollte.

Example 1:

```
tagmap:
mapping LOAD * inline [
a,b
Alpha,MyTag
Num,MyTag
];
tag fields using tagmap;
```

Example 2:

```
tag field Alpha with 'MyTag2';
```

Trace

Mit dem Befehl **trace** kann ein String im Fenster **Status der Skriptausführung** angezeigt und in die log-Datei geschrieben werden. Er eignet sich besonders zum Debugging. Wenn Sie die $-Erweiterungen von Variablen verwenden, die vor dem **trace**-Befehl berechnet werden, können Sie die Meldung anpassen.

Syntax:

```
Trace string
```

Example 1:

Der folgende Befehl kann direkt nach dem Ladebefehl verwendet werden, der die Haupttabelle lädt.

```
Trace Main table loaded;
```

Damit wird der Text „Main table loaded“ im Skriptausführungsdialogfeld und in der Protokolldatei angezeigt.

Example 2:

Die folgenden Befehle können direkt nach dem Ladebefehl verwendet werden, der die Haupttabelle lädt.

```
Let MyMessage = NoOfRows('Main') & ' rows in Main table';

Trace $(MyMessage);
```

Damit wird ein Text mit der Anzahl der Zeilen in Skriptausführungsdialogfeld und in der Protokolldatei angezeigt, beispielsweise „265,391 rows in Main table“.

Unmap

Mit dem Befehl **Unmap** wird der Feldwert Mapping deaktiviert, der mit dem Befehl **Map ... Using** für die anschließend geladenen Felder angegeben wurde.

Syntax:

```
Unmap *fieldlist
```

Argumente:

Argumente

Argument	Beschreibung
*fieldlist	Eine kommagetrennte Felderliste, für die das Mapping im Skript beendet werden soll. Das Sternchen * steht für alle Felder. Die Wildcards * und ? sind in Feldnamen zugelassen. Beim Gebrauch von Wildcards innerhalb von Feldnamen müssen diese gegebenenfalls in Anführungszeichen stehen.

Beispiele und Ergebnisse:

Beispiel	Ergebnis
Unmap Country;	Beendet das Mapping für das Feld Country.
Unmap A, B, C;	Beendet das Mapping für die Felder A, B und C.
Unmap * ;	Beendet das Mapping für alle Felder.

Unqualify

Die **Unqualify**-Anweisung wird verwendet, um die Qualifizierung von Feldnamen, die zuvor durch die **Qualify**-Anweisung aktiviert wurden, zu deaktivieren.

Syntax:

```
Unqualify *fieldlist
```

Argumente:

Argumente

Argument	Beschreibung
*fieldlist	Eine kommagetrennte Felderliste, für welche die Qualifizierung aktiviert werden sollte. Das Sternchen * steht für alle Felder. Die Wildcards * und ? sind in Feldnamen zugelassen. Beim Gebrauch von Wildcards innerhalb von Feldnamen müssen diese gegebenenfalls in Anführungszeichen stehen. Weitere Informationen über die Qualifizierung von Tabellen finden Sie in der Beschreibung des **Qualify**-Befehls.

Example 1:

Insbesondere in Datenbanken, deren Struktur nicht bekannt ist, möchten Sie möglicherweise nur Verknüpfungen über ein einziges oder wenige Felder zulassen. Dieser Vorgang wird im folgenden Beispiel dargestellt:

```
qualify *;
unqualify TransID;
SQL SELECT * from tab1;
SQL SELECT * from tab2;
SQL SELECT * from tab3;
```

Zuerst wird die Qualifizierung für alle Felder aktiviert.
Dann wird die Qualifizierung für **TransID** deaktiviert.
In diesem Fall werden die Tabellen **TransID**, *tab1* und *tab2* nur über das Feld *tab3* verknüpft. Alle anderen Felder werden mit dem Tabellennamen qualifiziert.

Untag

Dieser Skriptbefehl bietet eine Möglichkeit zum Entfernen von Tags aus Feldern oder Tabellen. Wird der Versuch unternommen, ein Tag aus einem Feld oder einer Tabelle zu entfernen, das bzw. die nicht in der App vorhanden ist, wird dieser Vorgang ignoriert.

Syntax:

```
Untag [field|fields] fieldlist with tagname
```

```
Untag [field|fields] fieldlist using mapname
```

```
Untag table tablelist with tagname
```

Argumente:

Argumente

Argument	Beschreibung
fieldlist	Ein oder mehrere Felder, aus denen Tags entfernt werden sollten, in einer kommagetrennten Liste.
mapname	Der Name einer Mapping-Tabelle, die durch die Mapping-Befehle **LOAD** oder **SELECT** geladen wurde.
tablelist	Eine kommagetrennte Liste der Tabellen, aus denen Tags entfernt werden sollten.
tagname	Der Name des Tags, das gelöscht wird.

Example 1:

```
tagmap:
mapping LOAD * inline [
a,b
Alpha,MyTag
Num,MyTag
];
Untag fields using tagmap;
```

Example 2:

```
Untag field Alpha with MyTag2;
```

3.4 Arbeitsverzeichnis

Wenn Sie in einem Skriptbefehl auf eine Datei verweisen ohne den Pfad anzugeben, sucht Qlik Sense die Datei in der folgenden Reihenfolge:

1. Das in einem **Directory**-Befehl spezifizierte Verzeichnis (wird nur im Legacymodus für die Skripterstellung unterstützt).
2. Wenn kein **Directory**-Befehl vorhanden ist, sucht Qlik Sense im Arbeitsverzeichnis.

Qlik Sense Desktop-Arbeitsverzeichnis

In Qlik Sense Desktop ist das Arbeitsverzeichnis *C:\Users\{user}\Documents\Qlik\Sense\Apps*.

Qlik Sense-Arbeitsverzeichnis

In einer Qlik Sense-Server-Installation wird das Arbeitsverzeichnis in Qlik Sense Repository Service angegeben, standardmäßig ist dies *C:\ProgramData\Qlik\Sense\Apps*. Weitere Informationen finden Sie in der Qlik Management Console-Hilfe.

4 Arbeiten mit Variablen im Dateneditor

Eine Variable in Qlik Sense ist eine Sammelbox, die einen statischen Wert oder eine Berechnung speichert, z. B. einen Zahlenwert oder alphanumerischen Wert. Wenn Sie die Variable in der App verwenden, wird jede Änderung an der Variable überall dort angewendet, wo die Variable verwendet wird. Sie können Variablen in der Variablenliste oder im Skript mithilfe des Dateneditors definieren. Den Wert einer Variablen legen Sie mithilfe der **Let**- oder **Set**-Anweisungen im Datenladeskript fest.

Sie können beim Bearbeiten eines Arbeitsblatts auch die Qlik Sense-Variablen aus der Variablenliste verwenden.

4.1 Überblick

Beginnt der Variablenwert mit einem Gleichheitszeichen '=', interpretiert Qlik Sense den Wert als Formel (Qlik Sense-Formel) und gibt das Ergebnis statt des eigentlichen Formeltexts zurück.

Beim Aufruf der Variablen im Skript wird diese durch den zugeordneten Wert ersetzt. Variablen können im Skript zur Dollarzeichenerweiterung sowie in verschiedenen Steuerungsbefehlen verwendet werden. Dies ist besonders nützlich, wenn der gleiche String, z. B. ein Pfad, häufig im Skript gebraucht wird.

Einige spezielle Systemvariablen werden von Qlik Sense am Beginn der Skriptausführung unabhängig von ihren vorherigen Werten festgelegt.

4.2 Festlegen einer Variable

Variablen bieten die Möglichkeit, statische Werte oder das Ergebnis einer Berechnung zu speichern. Zur Definition einer Variablen verwenden Sie die folgende Syntax:

```
set variablename = string
```

oder

```
let variable = expression
```

Der Befehl **Set** wird für die Stringzuweisung verwendet. Er weist den Text rechts vom Gleichheitszeichen der Variablen zu. Der Befehl **Let** wertet eine Formel rechts vom Gleichheitszeichen zur Laufzeit des Skripts aus und weist das Ergebnis der Formel der Variablen zu.

Bei Variablen wird zwischen Groß- und Kleinschreibung unterschieden.

Es wird nicht empfohlen, für eine Variable in Qlik Sense denselben Namen wie für ein Feld oder eine Funktion zu verwenden.

Beispiele:

Mit `set x = 3 + 4; //` erhält die Variable den String „3+4“ als Wert.

`let x = 3 + 4; //` gibt 7 als Wert zurück.

`set x = Today(); //` gibt „Today()“ als Wert zurück.

`let x = Today(); //` gibt das heutige Datum als Wert zurück, zum Beispiel „27.9.2021“.

4.3 Löschen einer Variablen

Wenn Sie eine Variable aus dem Skript entfernen und die Daten erneut laden, verbleibt die Variable in der App. Falls Sie die Variable vollständig aus der App entfernen möchten, müssen Sie sie auch über das Variablendialogfeld löschen.

4.4 Laden eines Variablenwerts als Feldwert

Wenn Sie einen Variablenwert als Feldwert in einem **LOAD**-Befehl laden möchten und das Ergebnis der Variablen Text – statt einer Zahl oder einer Formel – ist, müssen Sie die erweiterte Variable in einzelne Anführungszeichen setzen.

Beispiel:

In diesem Beispiel wird die Systemvariable mit der Liste der Skriptfehler in eine Tabelle geladen. Beachten Sie, dass die Erweiterung von ScriptErrorCount in der **If**-Klausel keine Anführungszeichen erfordert, während die Erweiterung von ScriptErrorList in Anführungszeichen gesetzt werden muss.

```
IF $(ScriptErrorCount) >= 1 THEN

   LOAD '$(ScriptErrorList)' AS Error AutoGenerate 1;
END IF
```

4.5 Variable Berechnung

Variablen mit berechneten Werten lassen sich in Qlik Sense vielfältig einsetzen, und das Ergebnis hängt von der Definition und der Art und Weise des Abrufs in einer Formel ab.

In diesem Beispiel werden einige Inline-Daten geladen:

```
LOAD * INLINE [
    Dim, Sales
    A, 150
    A, 200
    B, 240
    B, 230
    C, 410
    C, 330
];
```

Wir legen zwei Variablen fest:

```
Let vSales  = 'Sum(Sales)' ;
Let vSales2  = '=Sum(Sales)' ;
```

Bei der zweiten Variable wird ein Gleichheitszeichen vor der Formel hinzugefügt. Dadurch wird die Variable berechnet, bevor sie erweitert und die Formel evaluiert wird.

Wird die Variable vSales beispielsweise in einer Kennzahl alleine verwendet, ist das Ergebnis der String Sum (Sales), das heißt, es erfolgt keine Berechnung.

Wenn Sie ein Dollarzeichen hinzufügen und $(vSales) in der Formel abrufen, wird die Variable erweitert und die Summe von Sales angezeigt.

Wenn Sie $(vSales2) abrufen, wird die Variable vor ihrer Erweiterung berechnet. Dadurch wird als Ergebnis die Gesamtsumme von Sales angezeigt. Der Unterschied zwischen der Verwendung von =$(vSales) und =$(vSales2) als Kennzahlformeln wird in dieser Tabelle mit den Ergebnissen verdeutlicht:

Ergebnisse

Dim	$(vSales)	$(vSales2)
A	350	1560
B	470	1560
C	740	1560

Daraus ist ersichtlich, dass $(vSales) die Partialsumme eines Dimensionswerts ergibt, während $(vSales2) die Gesamtsumme liefert.

Die folgenden Skriptvariablen sind verfügbar:

- *Fehlervariablen (page 282)*
- *Variablen zur Interpretation von Zahlen (page 216)*
- *Systemvariablen (page 208)*
- *Variablen zur Verarbeitung der Werte (page 214)*

4.6 Systemvariablen

Systemvariablen, von denen einige systemdefiniert sind, bieten Informationen zum System und zur Qlik Sense-App.

Systemvariablen – Übersicht

Einige Funktionen werden nach der Übersicht genauer beschrieben. Bei diesen Funktionen können Sie auf den Funktionsnamen in der Syntax klicken, um direkt auf die Details zu der spezifischen Funktion zuzugreifen.

CreateSearchIndexOnReload

Diese Variable gibt an, ob die Suchindexdateien während des Neuladens der Daten erstellt werden sollen.

```
CreateSearchIndexOnReload
```

Floppy
Liefert die Laufwerksbezeichnung des ersten gefundenen Diskettenlaufwerks, in der Regel *a:*. Dies ist eine systemdefinierte Variable.

```
Floppy
```

Diese Variable wird im Standardmodus nicht unterstützt.

CD
Liefert die Laufwerksbezeichnung des ersten gefundenen CD-ROM-Laufwerks. Wird kein CD-ROM-Laufwerk gefunden, wird *c:* ausgegeben. Dies ist eine systemdefinierte Variable.

```
CD
```

Diese Variable wird im Standardmodus nicht unterstützt.

HidePrefix
Alle Felder, deren Namen mit dem bezeichneten Text beginnen, werden in gleicher Weise behandelt wie die Systemfelder. Dies ist eine benutzerdefinierte Variable.

```
HidePrefix
```

HideSuffix
Alle Felder, deren Namen mit dem bezeichneten Text enden, werden in gleicher Weise behandelt wie die Systemfelder. Dies ist eine benutzerdefinierte Variable.

```
HideSuffix
```

Include
Die **Include/Must_Include**-Variable spezifiziert eine Datei, die in das Skript mit einbezogen und als Skript-Code evaluiert werden sollte. Sie wird nicht zum Hinzufügen von Daten verwendet. Sie können Teile Ihres Script-Codes in einer separaten Textdatei speichern und in verschiedenen Apps verwenden. Dies ist eine benutzerdefinierte Variable.

```
$(Include=filename)
$(Must_Include=filename)
```

OpenUrlTimeout
Diese Variable definiert einen Timeout in Sekunden, den Qlik Sense für das Einlesen von Daten aus einer URL-Datenquelle einhalten sollte (z. B. HTML Seiten). Ist nichts definiert, ist der Timeout etwa 20 Minuten.

```
OpenUrlTimeout
```

QvPath
Gibt den Pfad zur Qlik Sense-Programmdatei zurück. Dies ist eine systemdefinierte Variable.

```
QvPath
```

Diese Variable wird im Standardmodus nicht unterstützt.

QvRoot
Gibt das Stammverzeichnis der Qlik Sense-Programmdatei zurück. Dies ist eine systemdefinierte Variable.

```
QvRoot
```

Diese Variable wird im Standardmodus nicht unterstützt.

QvWorkPath
Gibt den Pfad zur aktuellen Qlik Sense-App zurück. Dies ist eine systemdefinierte Variable.

```
QvWorkPath
```

Diese Variable wird im Standardmodus nicht unterstützt.

QvWorkRoot
Gibt das Stammverzeichnis der aktuellen Qlik Sense-App zurück. Dies ist eine systemdefinierte Variable.

```
QvWorkRoot
```

Diese Variable wird im Standardmodus nicht unterstützt.

StripComments
Hat diese Variable den Wert 0, bleibt die Bedeutung der Syntax /*..*/ bzw. // zur Kennzeichnung von Kommentaren im Skript nicht erhalten. Ist die Variable nicht definiert, bleibt die Bedeutung der Syntax zur Kennzeichnung von Kommentaren im Skript erhalten.

```
StripComments
```

Verbatim
Normalerweise werden vor dem Einlesen von Feldwerten in Qlik Sense führende oder nachfolgende Leerzeichen (ASCII-Wert 32) automatisch entfernt. Wird der Wert dieser Variablen auf 1 gesetzt, bleiben die Leerzeichen erhalten. Leerzeichen (ASCII 9) und geschützte Leerzeichen (ANSI 160) werden nie entfernt.

```
Verbatim
```

WinPath
Liefert den Pfad zu Windows. Dies ist eine systemdefinierte Variable.

```
WinPath
```

Diese Variable wird im Standardmodus nicht unterstützt.

WinRoot
Liefert das Root-Verzeichnis von Windows. Dies ist eine systemdefinierte Variable.

```
WinRoot
```

Diese Variable wird im Standardmodus nicht unterstützt.

CollationLocale
Gibt an, welches Gebietsschema für die Sortierreihenfolge und die Suchübereinstimmungen verwendet werden soll. Der Wert ist der Kulturname eines Gebietsschemas, z. B. 'en-US'.Dies ist eine systemdefinierte Variable.

```
CollationLocale
```

CreateSearchIndexOnReload

Diese Variable gibt an, ob die Suchindexdateien während des Neuladens der Daten erstellt werden sollen.

Syntax:

```
CreateSearchIndexOnReload
```

Sie können festlegen, ob die Suchindexdateien bereits während des Neuladens der Daten oder erst nach der ersten Suchanfrage des Benutzers erstellt werden sollen. Wenn Sie den Suchindex bereits während des Neuladens der Daten erstellen, ersparen Sie dem Benutzer längere Wartezeiten bei seiner ersten Suche. Wägen Sie diesen Vorteil gegen die eventuellen Nachteile einer längeren Dauer des Neuladevorgangs ab.

Wenn Sie diese Variable nicht berücksichtigen, werden die Suchindexdateien nicht während des Neuladens der Daten erstellt.

Für Sitzungs-Apps werden die Suchindexdateien niemals während des Neuladens erstellt, unabhängig von den Einstellungen dieser Variable.

Example 1: Erstellen der Suchindexfelder während des Neuladens der Daten

```
set CreateSearchIndexOnReload=1;
```

Example 2: Erstellen der Suchindexfelder nach dem ersten Suchvorgang

```
set CreateSearchIndexOnReload=0;
```

HidePrefix

Alle Felder, deren Namen mit dem bezeichneten Text beginnen, werden in gleicher Weise behandelt wie die Systemfelder. Dies ist eine benutzerdefinierte Variable.

Syntax:

```
HidePrefix
```

Beispiel:

```
set HidePrefix='_' ;
```

In diesem Beispiel werden alle Felder, deren Namen mit einem Unterstrich beginnen, wie Systemfelder behandelt, bzw. versteckt, d. h. sie erscheinen nur dann, wenn auch die Systemfelder angezeigt werden.

HideSuffix

Alle Felder, deren Namen mit dem bezeichneten Text enden, werden in gleicher Weise behandelt wie die Systemfelder. Dies ist eine benutzerdefinierte Variable.

Syntax:

```
HideSuffix
```

Beispiel:

```
set HideSuffix='%';
```

In diesem Beispiel werden alle Felder, deren Namen mit dem Prozentzeichen enden, wie Systemfelder behandelt bzw. versteckt, d. h. sie erscheinen nur dann, wenn auch die Systemfelder angezeigt werden.

Include

Die **Include/Must_Include**-Variable spezifiziert eine Datei, die in das Skript mit einbezogen und als Skript-Code evaluiert werden sollte. Sie wird nicht zum Hinzufügen von Daten verwendet. Sie können Teile Ihres Script-Codes in einer separaten Textdatei speichern und in verschiedenen Apps verwenden. Dies ist eine benutzerdefinierte Variable.

Diese Variable unterstützt im Standardmodus nur Ordner-Datenverbindungen.

Syntax:

```
$(Include=filename)
```

```
$(Must_Include=filename)
```

Es gibt zwei Versionen der Variable:

- **Include** generiert keinen Fehler, wenn die Datei nicht gefunden werden kann. Sie schlägt im Hintergrund fehl.
- **Must_Include** generiert einen Fehler, wenn die Datei nicht gefunden werden kann.

Wenn Sie keinen Pfad angeben, wird der Dateiname auf das Arbeitsverzeichnis der Qlik Sense-App bezogen. Sie können auch einen absoluten Dateipfad oder einen Pfad zu einer lib://-Ordnerverbindung angeben. Setzen Sie kein Leerzeichen vor oder nach dem Gleichheitszeichen.

*Die Konstruktion **set Include =**filename ist nicht möglich.*

Beispiele:

```
$(Include=abc.txt);
```

```
$(Must_Include=lib://DataFiles/abc.txt);
```

Beschränkungen

Eingeschränkte gegenseitige Kompatibilität zwischen UTF-8-codierten Dateien unter Windows und unter Linux.

Es ist optional, UTF-8 mit BOM (Byte Order Mark) zu verwenden. BOM kann die Nutzung von UTF-8 in Software beeinträchtigen, die keine Nicht-ASCII-Bytes am Beginn einer Datei erwartet, aber den Textstream ansonsten handhaben könnte.

- Windows-Systeme verwenden BOM in UTF-8, um zu identifizieren, dass eine Datei UTF-8-codiert ist, trotz des Umstands, dass keine Zweideutigkeit im Byte-Speicher vorhanden ist.
- Unix/Linux nutzen UTF-8 für Unicode, aber nicht BOM, da dies die Syntax für Befehlsdateien beeinflusst.

Dies hat verschiedene Auswirkungen für Qlik Sense.

- In Windows wird jede Datei, die mit einem UTF-8 BOM beginnt, als UTF-8-Skriptdatei betrachtet. Andernfalls wird ANSI-Codierung vorausgesetzt.
- In Linux ist die 8-Bit-Codeseite des Systemstandards UTF-8. Daher funktioniert UTF-8 auch ohne BOM.

Infolgedessen kann keine Übertragbarkeit garantiert werden. Es ist nicht immer möglich, eine Datei unter Windows zu erstellen, die von Linux interpretiert werden kann, und umgekehrt. Aufgrund der unterschiedlichen Handhabung von BOM besteht keine gegenseitige Kompatibilität zwischen den beiden Systemen hinsichtlich UTF-8-codierter Dateien.

OpenUrlTimeout

Diese Variable definiert einen Timeout in Sekunden, den Qlik Sense für das Einlesen von Daten aus einer URL-Datenquelle einhalten sollte (z. B. HTML Seiten). Ist nichts definiert, ist der Timeout etwa 20 Minuten.

Syntax:

```
OpenUrlTimeout
```

Beispiel:

```
set OpenUrlTimeout=10;
```

StripComments

Hat diese Variable den Wert 0, bleibt die Bedeutung der Syntax /*..*/ bzw. // zur Kennzeichnung von Kommentaren im Skript nicht erhalten. Ist die Variable nicht definiert, bleibt die Bedeutung der Syntax zur Kennzeichnung von Kommentaren im Skript erhalten.

Syntax:

```
StripComments
```

Einige Datenbanktreiber verwenden /*..*/ als Optimierungshinweise in **SELECT**-Befehlen. In diesem Fall sollten die Kommentare erst entfernt werden, nachdem der **SELECT**-Befehl an den Datenbanktreiber gesendet wurde.

Es wird empfohlen, die Variable nach dem betreffenden bzw. den betreffenden Befehl(en) auf 1 zurückzusetzen.

Beispiel:

```
set StripComments=0;
SQL SELECT * /* <optimization directive> */ FROM Table ;
set StripComments=1;
```

Verbatim

Normalerweise werden vor dem Einlesen von Feldwerten in Qlik Sense führende oder nachfolgende Leerzeichen (ASCII-Wert 32) automatisch entfernt. Wird der Wert dieser Variablen auf 1 gesetzt, bleiben die Leerzeichen erhalten. Leerzeichen (ASCII 9) und geschützte Leerzeichen (ANSI 160) werden nie entfernt.

Syntax:

```
Verbatim
```

Beispiel:

```
set Verbatim = 1;
```

4.7 Variablen zur Verarbeitung der Werte

In diesem Abschnitt werden die Variablen beschrieben, die zur Verarbeitung von NULL- und anderen Werten verwendet werden.

Variablen zur Verarbeitung der Werte – Übersicht

Jede Funktion wird nach der Übersicht genauer beschrieben. Sie können auch auf den Funktionsnamen in der Syntax klicken, um direkt auf die Details zu der spezifischen Funktion zuzugreifen.

NullDisplay
Das angegebene Symbol ersetzt alle NULL-Werte aus der ODBC-Datenquelle und Konnektoren auf dem niedrigsten Datenlevel. Dies ist eine benutzerdefinierte Variable.

```
NullDisplay
```

NullInterpret
Kommt das angegebene Symbol in einer Textdatei, NULL-Datei oder einem inline-Befehl vor, wird es als Excel interpretiert. Dies ist eine benutzerdefinierte Variable.

```
NullInterpret
```

NullValue
Wird der Befehl **NullAsValue** verwendet, ersetzt das festgelegte Symbol alle NULL-Werte in den mit **NullAsValue** angegebenen Feldern durch den angegebenen String.

```
NullValue
```

OtherSymbol
Definiert ein Symbol, das vor einem **LOAD/SELECT**-Befehl als "alle anderen Werte" interpretiert wird. Dies ist eine benutzerdefinierte Variable.

```
OtherSymbol
```

NullDisplay

Das angegebene Symbol ersetzt alle NULL-Werte aus der ODBC-Datenquelle und Konnektoren auf dem niedrigsten Datenlevel. Dies ist eine benutzerdefinierte Variable.

Syntax:

```
NullDisplay
```

Beispiel:

```
set NullDisplay='<NULL>';
```

NullInterpret

Kommt das angegebene Symbol in einer Textdatei, NULL-Datei oder einem inline-Befehl vor, wird es als Excel interpretiert. Dies ist eine benutzerdefinierte Variable.

Syntax:

```
NullInterpret
```

Beispiele:

```
set NullInterpret=' ';
set NullInterpret =;
```

Liefert für leere Werte in NULL keine Excel-Werte, jedoch für eine CSV-Textdatei.

```
set NullInterpret ='';
```

Liefert NULL-Werte für leere Werte in Excel.

NullValue

Wird der Befehl **NullAsValue** verwendet, ersetzt das festgelegte Symbol alle NULL-Werte in den mit **NullAsValue** angegebenen Feldern durch den angegebenen String.

Syntax:

```
NullValue
```

Beispiel:

```
NullAsValue Field1, Field2;
set NullValue='<NULL>';
```

OtherSymbol

Definiert ein Symbol, das vor einem **LOAD/SELECT**-Befehl als "alle anderen Werte" interpretiert wird. Dies ist eine benutzerdefinierte Variable.

Syntax:

```
OtherSymbol
```

Beispiel:

```
set OtherSymbol='+';
LOAD * inline
[X, Y
a, a
b, b];
LOAD * inline
[X, Z
a, a
+, c];
```

Der Feldwert Y='b' wird nun über das andere Symbol mit Z='c' verknüpft.

4.8 Variablen zur Interpretation von Zahlen

Variablen zur Interpretation von Zahlen sind vom System definiert. Die Variablen werden oben im Ladeskript hinzugefügt und wenden beim Ausführen des Skripts Zahlenformateinstellungen an. Diese Variablen können beliebig gelöscht, bearbeitet oder dupliziert werden.

Beim Erstellen einer neuen App werden die Variablen zur Interpretation von Zahlen entsprechend den aktuellen regionalen Einstellungen des Betriebssystems automatisch definiert. In Qlik Sense Desktop entspricht dies den Einstellungen des Computerbetriebssystems. In Qlik Sense sind sie entsprechend den Einstellungen des Betriebssystems auf dem Server, auf dem Qlik Sense installiert ist, festgelegt. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen

regionalen Einstellungen für Datums-, Zeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Währungsformat

MoneyDecimalSep
Das angegebene Dezimaltrennzeichen ersetzt das von den Regionaleinstellungen vorgegebene Dezimaltrennzeichen für Währungen.

```
MoneyDecimalSep
```

MoneyFormat
Das angegebene Symbol ersetzt das von den Regionaleinstellungen vorgegebene Währungssymbol.

```
MoneyFormat
```

MoneyThousandSep
Das angegebene Tausendertrennzeichen ersetzt das von den Regionaleinstellungen vorgegebene Tausendertrennzeichen für Währungen.

```
MoneyThousandSep
```

Zahlenformat

DecimalSep
Das angegebene Dezimaltrennzeichen ersetzt das von den Regionaleinstellungen vorgegebene Dezimaltrennzeichen.

```
DecimalSep
```

ThousandSep
Das angegebene Tausendertrennzeichen ersetzt das vom Betriebssystem vorgegebene Tausendertrennzeichen (Regionaleinstellungen).

```
ThousandSep
```

NumericalAbbreviation
Mit der numerischen Abkürzung wird festgelegt, welche Abkürzung für Größenordnungspräfixe von Zahlen verwendet werden, z. B. M für Mega oder eine Million (10^6) und µ für Mikro (10^{-6}).

```
NumericalAbbreviation
```

Zeitformat

DateFormat
Diese Umgebungsvariable definiert das Datumsformat, das als Standard in der App verwendet wird. Das Format wird zum Interpretieren und Formatieren von Datumsangaben verwendet. Wenn die Variable nicht definiert ist, wird bei der Skriptausführung das Datumsformat des Gebietsschemas des Betriebssystems abgerufen.

DateFormat

TimeFormat
Das angegebene Format für Uhrzeiten ersetzt das vom Betriebssystem vorgegebene Zeitformat (Regionaleinstellungen).

TimeFormat

TimestampFormat
Das angegebene Format für Zeitstempel (Datum und Uhrzeit) ersetzt das vom Betriebssystem vorgegebene (Regionaleinstellungen).

TimestampFormat

MonthNames
Das angegebene Format ersetzt das von den Regionaleinstellungen vorgegebene Format für Monatsnamen.

MonthNames

LongMonthNames
Das angegebene Format ersetzt das von den Regionaleinstellungen vorgegebene Format für ausgeschriebene Monatsnamen.

LongMonthNames

DayNames
Das angegebene Format ersetzt das von den Regionaleinstellungen vorgegebene Format für Wochentage.

DayNames

LongDayNames
Das angegebene Format ersetzt das von den Regionaleinstellungen vorgegebene Format für ausgeschriebene Wochentage.

LongDayNames

FirstWeekDay
Ganzzahlwert, der definiert, welcher Tag als erster Tag der Woche verwendet wird.

FirstWeekDay

BrokenWeeks
Diese Einstellung definiert, ob Wochen gestückelt werden oder nicht.

BrokenWeeks

ReferenceDay
Die Einstellung legt fest, welcher Tag im Januar als Referenztag Woche 1 definiert.

ReferenceDay

FirstMonthOfYear
Die Einstellung legt fest, welcher Monat als erster Monat des Jahres verwendet wird. So kann der erste Monat eines Geschäftsjahres festgelegt werden, z. B. 1. April.

Diese Einstellung wird zurzeit nicht verwendet, ist jedoch für die Zukunft reserviert.

Gültige Einstellungen sind vom 1 (Januar) bis 12 (Dezember). Standardformat ist 1.

Syntax:
```
FirstMonthOfYear
```

Beispiel:

```
Set FirstMonthOfYear=4; //Sets the year to start in April
```

BrokenWeeks

Diese Einstellung definiert, ob Wochen gestückelt werden oder nicht.

Syntax:
```
BrokenWeeks
```
In Qlik Sense werden die regionalen Einstellungen bei der Erstellung der App abgerufen, und die entsprechenden Einstellungen werden im Skript als Umgebungsvariablen gespeichert.

Ein nordamerikanischer App-Entwickler erhält oft `Set BrokenWeeks=1;` im Skript, was gestückelten Wochen entspricht. Ein europäischer App-Entwickler erhält oft `Set BrokenWeeks=0;` im Skript, was nicht gestückelten Wochen entspricht.

Nicht gestückelte Wochen bedeutet Folgendes:

- In manchen Jahren beginnt die Woche 1 im Dezember und in anderen Jahren reicht die letzte Woche des Vorjahres bis in den Januar hinein.
- Gemäß ISO 8601 hat die Woche 1 immer mindestens 4 Tage im Januar. In Qlik Sense kann dies anhand der Variablen `ReferenceDay` konfiguriert werden.

Gestückelte Wochen bedeutet Folgendes:

- Die letzte Woche des Jahres reicht nie bis in den Januar hinein.
- Die Woche 1 beginnt am 1. Januar und ist in den meisten Fällen keine ganze Woche.

Folgende Werte können verwendet werden:

- 0 (= nicht gestückelte Wochen verwenden)
- 1 (= gestückelte Wochen verwenden)

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer

Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Beispiele:

Wenn Sie ISO-Einstellungen für Wochen und Wochennummern verwenden möchten, müssen Sie Folgendes in das Skript einschließen:

```
Set FirstWeekDay=0;
Set BrokenWeeks=0;   //(use unbroken weeks)
Set ReferenceDay=4;
```

Wenn Sie US-Einstellungen verwenden möchten, müssen Sie Folgendes in das Skript einschließen:

```
Set FirstWeekDay=6;
Set BrokenWeeks=1;   //(use broken weeks)
Set ReferenceDay=1;
```

DateFormat

Diese Umgebungsvariable definiert das standardmäßig in der App verwendete Datumsformat und dasjenige und von Funktionen, die ein Datum zurückgeben (z. B. `date()` und `date#()`). Das Format wird zum Interpretieren und Formatieren von Datumsangaben verwendet. Wenn die Variable nicht definiert ist, wird das von Ihren regionalen Einstellungen festgelegte Datumsformat abgerufen, wenn das Skript ausgeführt wird.

Syntax:

```
DateFormat
```

Beispiele für die Funktion „DateFormat“

Beispiel	Ergebnis
`Set DateFormat='M/D/YY'; //(US format)`	Bei dieser Verwendung der Funktion `DateFormat` wird das Datum im US-Format (Monat/Tag/Jahr) definiert.
`Set DateFormat='DD/MM/YY'; //(UK date format)`	Bei dieser Verwendung der Funktion `DateFormat` wird das Datum im UK-Format (Tag/Monat/Jahr) definiert.
`Set DateFormat='YYYY/MM/DD'; //(ISO date format)`	Bei dieser Verwendung der Funktion `DateFormat` wird das Datum im ISO-Format (Jahr/Monat/Tag) definiert.

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Beispiel 1 – Systemvariablenstandard

Ladeskript und Ergebnisse

Überblick

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz mit Datumsangaben.
- Die Funktion `DateFormat`, die das US-Datumsformat verwendet.

In diesem Beispiel wird ein Datensatz in eine Tabelle namens „`Transactions`“ geladen. Sie enthält ein Feld `date`. Die US-Definition für `DateFormat` wird verwendet. Dieses Muster wird für implizite Text-zu-Datum-Konvertierung verwendet, wenn die Text-Datumsangaben geladen werden.

Ladeskript

```
Set DateFormat='MM/DD/YYYY';

Transactions:
LOAD
date,
month(date) as month,
id,
amount
INLINE
[
date,id,amount
01/01/2022,1,1000
02/01/2022,2,2123
03/01/2022,3,4124
04/01/2022,4,2431
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `month`

Erstellen Sie die folgende Kennzahl:

```
=sum(amount)
```

Ergebnistabelle

date	Monat	=sum(amount)
01/01/2022	Jan	1000
02/01/2022	Feb	2123
03/01/2022	Mär	4124
04/01/2022	Apr	2431

Die `DateFormat`-Definition MM/TT/JJJJ wird für die implizite Konvertierung von Text in Datum verwendet, daher wird das Feld `date` korrekt als ein Datum interpretiert. Das gleiche Format wird zum Anzeigen des Datums verwendet, wie in der Ergebnistabelle gezeigt.

Beispiel 2 – Systemvariable ändern

Ladeskript und Ergebnisse

Überblick

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Der gleiche Datensatz aus dem vorigen Beispiel.
- Die Funktion `DateFormat`, die das Format „TT/MM/JJJJ“ verwendet.

Ladeskript

```
SET DateFormat='DD/MM/YYYY';
Transactions:
LOAD
date,
month(date) as month,
id,
amount
INLINE
[
date,id,amount
01/01/2022,1,1000
```

```
02/01/2022,2,2123
03/01/2022,3,4124
04/01/2022,4,2431
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `month`

Erstellen Sie die folgende Kennzahl:

```
=sum(amount)
```

Ergebnistabelle

date	**Monat**	**=sum(amount)**
01/01/2022	Jan	1000
02/01/2022	Jan	2123
03/01/2022	Jan	4124
04/01/2022	Jan	2431

Da die `DateFormat`-Definition auf „TT/MM/JJJJ“ festgelegt wurde, sehen Sie, dass die zwei Ziffern nach dem ersten Symbol „/“ als Monat interpretiert wurden. Daher stammen alle Datensätze aus dem Monat Januar.

Beispiel 3 – Datumsinterpretation

Ladeskript und Ergebnisse

Überblick

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz mit Datumsangaben in numerischem Format.
- Die Variable `DateFormat`, die das Format „TT/MM/JJJJ“ verwendet.
- Die Variable `date()`.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
date(numerical_date),
month(date(numerical_date)) as month,
id,
```

```
amount
Inline
[
numerical_date,id,amount
43254,1,1000
43255,2,2123
43256,3,4124
43258,4,2431
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `month`

Erstellen Sie die folgende Kennzahl:

```
=sum(amount)
```

Ergebnistabelle

date	**Monat**	**=sum(amount)**
06/03/2022	Jun	1000
06/04/2022	Jun	2123
06/05/2022	Jun	4124
06/07/2022	Jun	2431

Im Ladeskript verwenden Sie die Funktion `date()` zum Konvertieren des numerischen Datums in ein Datumsformat. Da Sie kein spezifisches Format als zweites Argument in der Funktion angeben, wird das Format `DateFormat` verwendet. Dies ergibt das Datumsfeld mit dem Format „MM/TT/JJJJ“.

Beispiel 4 – Formatierung im Format eines anderen Landes

Ladeskript und Ergebnisse

Überblick

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Einen Datensatz mit Datumsangaben.
- Die Variable `DateFormat`, die das Format „TT/MM/JJJJ“ verwendet, aber durch Schrägstriche auskommentiert wird.

Ladeskript

```
// SET DateFormat='DD/MM/YYYY';


Transactions:
Load
date,
month(date) as month,
id,
amount
Inline
[
date,id,amount
22-05-2022,1,1000
23-05-2022,2,2123
24-05-2022,3,4124
25-05-2022,4,2431
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `month`

Erstellen Sie die folgende Kennzahl:

```
=sum(amount)
```

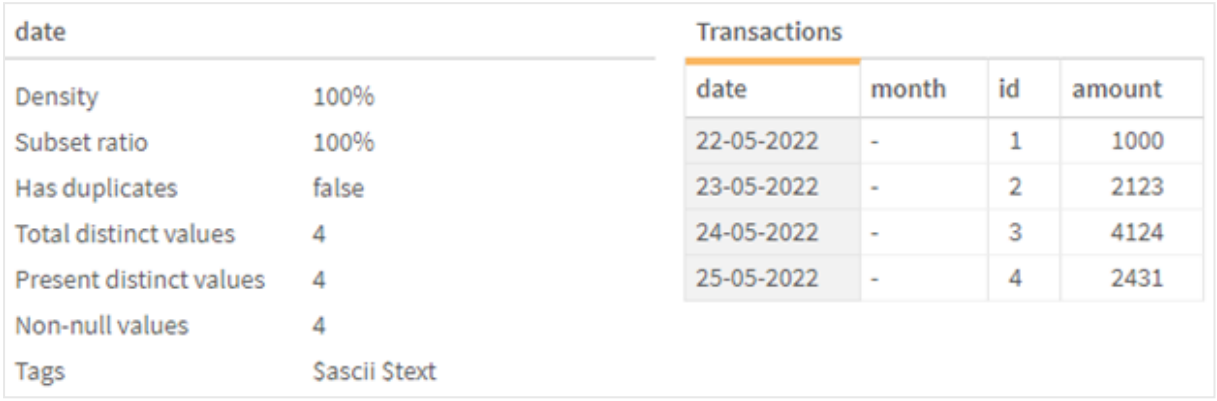
Ergebnistabelle

date	**Monat**	**=sum(amount)**
22-05-2022	-	1000
23-05-2022	-	2123
24-05-2022	-	4124
25-05-2022	-	2431

Im anfänglichen Ladeskript wird als `DateFormat` das standardmäßige „MM/TT/JJJJ“ verwendet. Da das Feld `date` im Transaktionsdatensatz nicht dieses Format aufweist, wird das Feld nicht als Datum interpretiert. Dies zeigt sich in der Ergebnistabelle, wo die Werte des Felds `month` null sind.

Sie können die interpretierten Datentypen in der Datenmodellansicht überprüfen, indem Sie die „Tags“-Eigenschaften des Felds `date` prüfen:

Vorschau der Tabelle `Transactions`. Beachten Sie, dass die „Tags“ für das Feld `date` darauf hinweisen, dass die Texteingabedaten nicht implizit in Datum/Zeitstempel konvertiert wurden.



date

Density	100%
Subset ratio	100%
Has duplicates	false
Total distinct values	4
Present distinct values	4
Non-null values	4
Tags	$ascii $text

Transactions

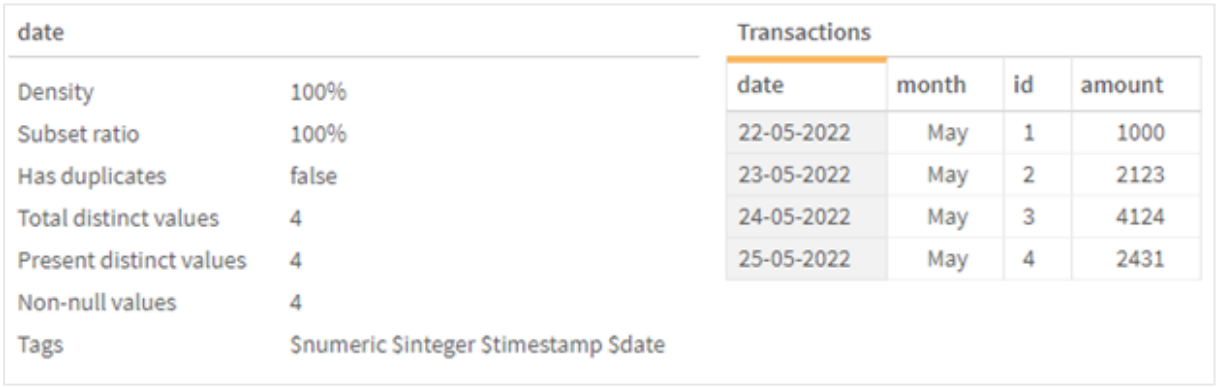
date	month	id	amount
22-05-2022	-	1	1000
23-05-2022	-	2	2123
24-05-2022	-	3	4124
25-05-2022	-	4	2431

Dies kann durch Aktivieren der Systemvariaben `DateFormat` behoben werden:

```
// SET DateFormat='DD/MM/YYYY';
```

Entfernen Sie den doppelten Schrägstrich und laden Sie die Daten erneut.

Vorschau der Tabelle `Transactions`. Beachten Sie, dass die „Tags“ für das Feld `date` darauf hinweisen, dass die Texteingabedaten implizit in Datum/Zeitstempel konvertiert wurden.



date

Density	100%
Subset ratio	100%
Has duplicates	false
Total distinct values	4
Present distinct values	4
Non-null values	4
Tags	$numeric $integer $timestamp $date

Transactions

date	month	id	amount
22-05-2022	May	1	1000
23-05-2022	May	2	2123
24-05-2022	May	3	4124
25-05-2022	May	4	2431

DayNames

Das angegebene Format ersetzt das von den Regionaleinstellungen vorgegebene Format für Wochentage.

Syntax:

`DayNames`

Wenn die Variable geändert wird, ist ein Semikolon (;) erforderlich, um die einzelnen Werte zu trennen.

Beispiele der Funktion „DayName“

Funktionsbeispiel	Ergebnisdefinition
`Set DayNames='Mon;Tue;Wed;Thu;Fri;Sat;Sun';`	Diese Verwendung der Funktion „DayNames“ definiert die Namen von Tagen in ihrer abgekürzten Form.

Funktionsbeispiel	Ergebnisdefinition
`Set DayNames='M;Tu;W;Th;F;Sa;Su';`	Diese Verwendung der Funktion „DayNames“ definiert die Namen von Tagen nach ihren Anfangsbuchstaben.

Die Funktion `DayNames` wird oft in Kombination mit den folgenden Funktionen verwendet:

Verwandte Funktionen

Funktion	Interaktion
weekday (page 1097)	Skriptfunktion zum Zurückgeben von `DayNames` als Feldwerten.
Date (page 1258)	Skriptfunktion zum Zurückgeben von `DayNames` als Feldwerte.
LongDayNames (page 238)	Langformwerte von `DayNames`.

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Beispiel 1 – Systemvariablenstandard

Ladeskript und Ergebnisse

Übersicht

In diesem Beispiel wird jedes Datum im Datensatz im Format MM/TT/JJJJ festgelegt.

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz mit Datumsangaben, der in eine Tabelle namens `Transactions` geladen wird.
- Ein Feld `date`.
- Die Standarddefinition `DayNames`.

Ladeskript

```
SET DayNames='Mon;Tue;Wed;Thu;Fri;Sat;Sun';
```

```
Transactions:
LOAD
date,
WeekDay(date) as dayname,
id,
amount
INLINE
[
date,id,amount
01/01/2022,1,1000
02/01/2022,2,2123
03/01/2022,3,4124
04/01/2022,4,2431
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `dayname`

Erstellen Sie die folgende Kennzahl:

```
sum(amount)
```

Ergebnistabelle

date	dayname	sum(amount)
01/01/2022	Sa	1000
02/01/2022	Di	2123
03/01/2022	Di	4124
04/01/2022	Fr	2431

Im Ladeskript wird die Funktion `WeekDay` mit dem Feld `date` als bereitgestelltes Argument verwendet. In der Ausgabetabelle zeigt die Ausgabe dieser Funktion `WeekDay` die Tage der Woche im Format der Definition `DayNames` an.

Beispiel 2 – Systemvariable ändern

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein. Es werden derselbe Datensatz und dasselbe Szenario wie im ersten Beispiel verwendet.

Am Beginn des Skripts wird die Definition `DayNames` aber geändert, um die abgekürzten Wochentage in Afrikaans zu verwenden.

Ladeskript

```
SET DayNames='Ma;Di;Wo;Do;Vr;Sa;So';

Transactions:
Load
date,
WeekDay(date) as dayname,
id,
amount
Inline
[
date,id,amount
01/01/2022,1,1000
02/01/2022,2,2123
03/01/2022,3,4124
04/01/2022,4,2431
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `dayname`

Erstellen Sie die folgende Kennzahl:

```
sum(amount)
```

Ergebnistabelle

date	dayname	sum(amount)
01/01/2022	Sa	1000
02/01/2022	Di	2123
03/01/2022	Di	4124
04/01/2022	Vr	2431

In der Ausgabetabelle zeigt die Ausgabe dieser Funktion `weekDay` die Tage der Woche im Format der Definition `DayNames` an.

Beachten Sie: Wenn die Sprache für `DayNames` wie in diesem Beispiel geändert wird, enthält `LongDayNames` nach wie vor die Wochentage auf Englisch. Dies muss ebenfalls geändert werden, wenn beide Variablen in der Anwendung verwendet werden.

Beispiel 3 – Datumsfunktion

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz mit Datumsangaben, der in eine Tabelle namens `Transactions` geladen wird.
- Ein Feld `date`.
- Die Standarddefinition `DayNames`.

Ladeskript

```
SET DayNames='Mon;Tue;Wed;Thu;Fri;Sat;Sun';

Transactions:
Load
date,
Date(date,'WWW') as dayname,
id,
amount
Inline
[
date,id,amount
01/01/2022,1,1000
02/01/2022,2,2123
03/01/2022,3,4124
04/01/2022,4,2431
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `dayname`

Erstellen Sie die folgende Kennzahl:

```
sum(amount)
```

Ergebnistabelle

date	dayname	sum(amount)
01/01/2022	Sa	1000
02/01/2022	Di	2123
03/01/2022	Di	4124
04/01/2022	Fr	2431

Die Standarddefinition DayNames wird verwendet. Im Ladeskript wird die Funktion Date mit dem Feld date als erstes Argument verwendet. Das zweite Argument ist WWW. Diese Formatierung konvertiert das Ergebnis in die Werte, die in der Definition DayNames gespeichert sind. Dies wird in der Ausgabe der Ergebnistabelle angezeigt.

DecimalSep

Das angegebene Dezimaltrennzeichen ersetzt das von den Regionaleinstellungen vorgegebene Dezimaltrennzeichen.

Qlik Sense interpretiert Text automatisch als Zahlen, wenn ein erkennbares Zahlenmuster gefunden wird. Die Systemvariablen ThousandSep und DecimalSep bestimmen die Zusammensetzung der angewendeten Muster, wenn Text als Zahlen analysiert wird. Die Variablen ThousandSep und DecimalSep legen das Muster des Standardzahlenformats fest, wenn Zahleninhalte in Frontend-Diagrammen und -Tabellen visualisiert werden. Sie wirken sich somit direkt auf die Optionen für **Zahlenformat** für alle Frontend-Formeln aus.

Wenn das Tausendertrennzeichen ein „,“ und das Dezimaltrennzeichen ein „.“ ist, werden die Muster der folgenden Beispiele implizit in die entsprechenden numerischen Werte konvertiert:

```
0,000.00
```

```
0000.00
```

```
0,000
```

Dies sind Beispiele von Mustern, die als Text unverändert bleiben und nicht in Zahlen konvertiert werden:

```
0.000,00
```

```
0,00
```

Syntax:

DecimalSep

Funktionsbeispiele

Beispiel	Ergebnis
`Set DecimalSep='.';`	Legt „.“ als Dezimaltrennzeichen fest.
`Set DecimalSep=',';`	Legt „,“ als Dezimaltrennzeichen fest.

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung SET DateFormat in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für

Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Beispiel – Auswirkung der Festlegung von Zahlentrennzeichen-Variablen auf verschiedene Eingabedaten

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz von Summen und Datumsangaben, in dem die Summen in verschiedenen Formatmustern angegeben sind.
- Eine Tabelle namens `Transactions`.
- Die Variable `DecimalSep`, die auf „.“ festgelegt ist.
- Die Variable `ThousandSep`, die auf „,“ festgelegt ist.
- Die Variable `delimiter`, die auf das Zeichen „|“ festgelegt ist, um die einzelnen Felder in einer Zeile zu trennen.

Ladeskript

```
Set ThousandSep=',';
Set DecimalSep='.';

Transactions:
Load date,
id,
amount as amount
Inline
[
date|id|amount
01/01/2022|1|1.000-45
01/02/2022|2|23.344
01/03/2022|3|4124,35
01/04/2022|4|2431.36
01/05/2022|5|4,787
01/06/2022|6|2431.84
01/07/2022|7|4132.5246
01/08/2022|8|3554.284
01/09/2022|9|3.756,178
01/10/2022|10|3,454.356
] (delimiter is '|');
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension `amount` hinzu.

Erstellen Sie die folgende Kennzahl:

```
=sum(amount)
```

Ergebnistabelle

Amount	**=Sum(amount)**	
Summen		20814.7086
1.000-45		
3.756,178		
4124,35		
	23.344	23.344
	2431.36	2431.36
	2431.84	2431.84
	3,454.356	3454.356
	3554.284	3554.284
	4132.5246	4132.5246
	4,787	4787

Jeder Wert, der nicht als Zahl interpretiert wird, behält das Textformat bei und wird standardmäßig linksbündig ausgerichtet. Alle erfolgreich konvertierten Werte werden rechtsbündig ausgerichtet und behalten das ursprüngliche Eingabeformat bei.

Die Formelspalte zeigt die numerische Entsprechung, die standardmäßig nur mit einem Dezimaltrennzeichen „.“ formatiert ist. Dies kann in der Formelkonfiguration mit der Dropdown-Einstellung **Zahlenformat** überschrieben werden.

FirstWeekDay

Ganzzahlwert, der definiert, welcher Tag als erster Tag der Woche verwendet wird.

Syntax:

```
FirstWeekDay
```

Montag ist der erste Tag der Woche gemäß ISO 8601, dem internationalen Standard für die Darstellung von Datumsangaben und Uhrzeiten. Montag wird auch in einer Reihe von Ländern als erster Tag der Woche verwendet, beispielsweise in Großbritannien, Frankreich, Deutschland und Schweden.

In anderen Ländern wie den USA und Kanada wird jedoch Sonntag als Wochenbeginn betrachtet.

In Qlik Sense werden die regionalen Einstellungen bei der Erstellung der App abgerufen, und die entsprechenden Einstellungen werden im Skript als Umgebungsvariablen gespeichert.

Ein nordamerikanischer App-Entwickler erhält oft `Set FirstWeekDay=6;` im Skript, was Sonntag entspricht. Ein europäischer App-Entwickler erhält oft `Set FirstWeekDay=0;` im Skript, was Montag entspricht.

Werte, die als FirstWeekDay festgelegt werden können

Wert	Tag
0	Montag
1	Dienstag
2	Mittwoch
3	Donnerstag
4	Freitag
5	Samstag
6	Sonntag

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Beispiele:

Wenn Sie ISO-Einstellungen für Wochen und Wochennummern verwenden möchten, müssen Sie Folgendes in das Skript einschließen:

```
Set FirstWeekDay=0;  // Monday as first week day
Set BrokenWeeks=0;
Set ReferenceDay=4;
```

Wenn Sie US-Einstellungen verwenden möchten, müssen Sie Folgendes in das Skript einschließen:

```
Set FirstWeekDay=6;  // Sunday as first week day
Set BrokenWeeks=1;
Set ReferenceDay=1;
```

Beispiel 1 – Verwendung des Standardwerts (Skript)

Ladeskript und Ergebnisse

Überblick

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

In diesem Beispiel verwendet das Ladeskript den Standardwert der Qlik Sense-Systemvariablen `FirstWeekDay=6`. Diese Daten enthalten Angaben für die ersten 14 Tage des Jahres 2020.

Ladeskript

```
// Example 1: Load Script using the default value of FirstWeekDay=6, i.e. Sunday

SET FirstWeekDay = 6;

Sales:
LOAD
        date,
        sales,
        week(date) as week,
        weekday(date) as weekday
Inline [
date,sales
01/01/2021,6000
01/02/2021,3000
01/03/2021,6000
01/04/2021,8000
01/05/2021,5000
01/06/2020,7000
01/07/2020,3000
01/08/2020,5000
01/09/2020,9000
01/10/2020,5000
01/11/2020,7000
01/12/2020,7000
01/13/2020,7000
01/14/2020,7000
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `week`

- weekday

Ergebnistabelle

Datum	Woche	weekday
01/01/2021	1	Mi
01/02/2021	1	Do
01/03/2021	1	Fr
01/04/2021	1	Sa
01/05/2021	2	So
01/06/2020	2	Mo
01/07/2020	2	Di
01/08/2020	2	Mi
01/09/2020	2	Do
01/10/2020	2	Fr
01/11/2020	2	Sa
01/12/2020	3	So
01/13/2020	3	Mo
01/14/2020	3	Di

Da die Standardeinstellungen verwendet werden, ist die `FirstWeekDay`-Systemvariable auf 6 festgelegt. In der Ergebnistabelle sehen Sie, dass jede neue Woche an einem Sonntag beginnt (der 5. und 12. Januar).

Beispiel 2 – Ändern der Variablen FirstWeekDay (Skript)

Ladeskript und Ergebnisse

Überblick

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

In diesem Beispiel enthalten die Daten Angaben für die ersten 14 Tage des Jahres 2020. Am Anfang des Skripts legen wir die Variable `FirstWeekDay` auf 3 fest.

Ladeskript

```
// Example 2: Load Script setting the value of FirstWeekDay=3, i.e. Thursday

SET FirstWeekDay = 3;

Sales:
LOAD
       date,
```

```
       sales,
       week(date) as week,
       weekday(date) as weekday
Inline [
date,sales
01/01/2021,6000
01/02/2021,3000
01/03/2021,6000
01/04/2021,8000
01/05/2021,5000
01/06/2020,7000
01/07/2020,3000
01/08/2020,5000
01/09/2020,9000
01/10/2020,5000
01/11/2020,7000
01/12/2020,7000
01/13/2020,7000
01/14/2020,7000
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- date
- week
- weekday

Ergebnistabelle

Datum	Woche	weekday
01/01/2021	52	Mi
01/02/2021	1	Do
01/03/2021	1	Fr
01/04/2021	1	Sa
01/05/2021	1	So
01/06/2020	1	Mo
01/07/2020	1	Di
01/08/2020	1	Mi
01/09/2020	2	Do
01/10/2020	2	Fr
01/11/2020	2	Sa

Datum	Woche	weekday
01/12/2020	2	So
01/13/2020	2	Mo
01/14/2020	2	Di

Da die `FirstWeekDay` -Systemvariable auf 3 festgelegt ist, ist der erste Tag jeder Woche ein Donnerstag. In der Ergebnistabelle sehen Sie, dass jede neue Woche mit einem Donnerstag beginnt (2. und 9. Januar).

LongDayNames

Das angegebene Format ersetzt das von den Regionaleinstellungen vorgegebene Format für ausgeschriebene Wochentage.

Syntax:

```
LongDayNames
```

Das folgende Beispiel der `LongDayNames`-Funktion definiert Tagesnamen vollumfänglich:

```
Set LongDayNames='Monday;Tuesday;Wednesday;Thursday;Friday;Saturday;Sunday';
```

Wenn die Variable geändert wird, ist ein Semikolon (`;`) erforderlich, um die einzelnen Werte zu trennen.

Die Funktion `LongDayNames` kann in Kombination mit der Funktion *Date (page 1258)* verwendet werden, wodurch `DayNames` als Feldwerte zurückgegeben wird.

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Beispiel 1 – Systemvariablenstandard

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz mit Datumsangaben, der in eine Tabelle namens `Transactions` geladen wird.
- Ein Feld `date`.
- Die Standarddefinition `LongDayNames`.

Ladeskript

```
SET LongDayNames='Monday;Tuesday;Wednesday;Thursday;Friday;Saturday;Sunday';

Transactions:
LOAD
date,
Date(date,'WWWW') as dayname,
id,
amount
INLINE
[
date,id,amount
01/01/2022,1,1000
02/01/2022,2,2123
03/01/2022,3,4124
04/01/2022,4,2431
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `dayname`

Erstellen Sie die folgende Kennzahl:

```
=sum(amount)
```

Ergebnistabelle

date	dayname	=sum(amount)
01/01/2022	Samstag	1000
02/01/2022	Dienstag	2123
03/01/2022	Dienstag	4124
04/01/2022	Freitag	2431

Im Ladeskript wird zum Erstellen eines Felds mit dem Namen `dayname` die Funktion `Date` mit dem Feld `date` als erstem Argument verwendet. Das zweite Argument in der Funktion ist die Formatierung `WWWW`.

Anhand dieser Formatierung werden die Werte aus dem ersten Argument in den entsprechenden vollständigen Tagesnamen konvertiert, der in der Variablen `LongDayNames` festgelegt ist. In der Ergebnistabelle wird dies in den Feldwerten des neu erstellten Felds `dayname` gezeigt.

Beispiel 2 – Systemvariable ändern

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Es werden derselbe Datensatz und dasselbe Szenario wie im ersten Beispiel verwendet. Am Beginn des Skripts wird die Definition `LongDayNames` aber geändert, um die Wochentage auf Spanisch zu verwenden.

Ladeskript

```
SET LongDayNames='Lunes;Martes;Miércoles;Jueves;Viernes;Sábado;Domingo';

Transactions:
LOAD
date,
Date(date,'WWWW') as dayname,
id,
amount
INLINE
[
date,id,amount
01/01/2022,1,1000
02/01/2022,2,2123
03/01/2022,3,4124
04/01/2022,4,2431
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `dayname`

Erstellen Sie die folgende Kennzahl:

```
=sum(amount)
```

Ergebnistabelle

date	dayname	=sum(amount)
01/01/2022	Sábado	1000
02/01/2022	Martes	2123
03/01/2022	Martes	4124
04/01/2022	Viernes	2431

Im Ladeskript wird die Variable LongDayNames geändert, um die Wochentage auf Spanisch aufzulisten.

Dann erstellen Sie ein Feld mit dem Namen dayname. Dies ist die verwendete Funktion Date mit dem Feld date als erstem Argument.

Das zweite Argument in der Funktion ist die Formatierung WWWW. Anhand dieser Formatierung Qlik Sense werden die Werte aus dem ersten Argument in den entsprechenden vollständigen Tagesnamen konvertiert, der in der Variablen LongDayNames festgelegt ist.

In der Ergebnistabelle zeigen die Feldwerte des erstellten Felds dayname die vollständig ausgeschriebenen Wochentage auf Spanisch an.

LongMonthNames

Das angegebene Format ersetzt das von den Regionaleinstellungen vorgegebene Format für ausgeschriebene Monatsnamen.

Syntax:

LongMonthNames

Wenn die Variable geändert wird, ist ein ; erforderlich, um die einzelnen Werte zu trennen.

Das folgende Beispiel der Funktion „LongMonthNames“ definiert Tagesnamen in ausgeschriebener Form:

```
Set
LongMonthNames='January;February;March;April;May;June;July;August;September;October;November;D
ecember';
```

Die Funktion LongMonthNames wird oft in Kombination mit den folgenden Funktionen verwendet:

Verwandte Funktionen

Funktion	Interaktion
Date (page 1258)	Skriptfunktion zum Zurückgeben von DayNames als Feldwerten.
LongDayNames (page 238)	Langformwerte von DayNames.

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung SET DateFormat in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Beispiel 1 – Systemvariablenstandard

Ladeskript und Ergebnisse

Überblick

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz mit Datumsangaben, die in eine Tabelle mit dem Namen `Transactions` geladen werden.
- Ein Feld `date`.
- Die Standarddefinition `LongMonthNames`.

Ladeskript

```
SET
LongMonthNames='January;February;March;April;May;June;July;August;September;October;November;D
ecember';

Transactions:
Load
date,
Date(date,'MMMM') as monthname,
id,
amount
Inline
[
date,id,amount
01/01/2022,1,1000.45
01/02/2022,2,2123.34
01/03/2022,3,4124.35
01/04/2022,4,2431.36
01/05/2022,5,4787.78
01/06/2022,6,2431.84
01/07/2022,7,2854.83
01/08/2022,8,3554.28
01/09/2022,9,3756.17
01/10/2022,10,3454.35
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `monthname`

Erstellen Sie die folgende Kennzahl:

```
=sum(amount)
```

Ergebnistabelle

date	monthname	sum(amount)
01/01/2022	Januar	1000.45
01/02/2022	Januar	2123.34
01/03/2022	Januar	4124.35
01/04/2022	Januar	2431.36
01/05/2022	Januar	4787.78
01/06/2022	Januar	2431.84
01/07/2022	Januar	2854.83
01/08/2022	Januar	3554.28
01/09/2022	Januar	3756.17
01/10/2022	Januar	3454.35

Die Standarddefinition `LongMonthNames` wird verwendet. Im Ladeskript wird zum Erstellen eines Felds mit dem Namen `month` die Funktion `Date` mit dem Feld `date` als erstem Argument verwendet. Das zweite Argument in der Funktion ist die Formatierung `MMMM`.

Anhand dieser Formatierung Qlik Sense werden die Werte aus dem ersten Argument in den entsprechenden vollständigen Monatsnamen konvertiert, der in der Variablen `LongMonthNames` festgelegt ist. In der Ergebnistabelle wird dies in den Feldwerten des neu erstellten Felds `month` gezeigt.

Beispiel 2 – Systemvariable ändern

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Einen Datensatz mit Datumsangaben, die in eine Tabelle mit dem Namen `Transactions` geladen werden.
- Ein Feld `date`.
- Die Variable `LongMonthNames` wird geändert, um die abgekürzten Wochentage auf Spanisch zu verwenden.

Ladeskript

```
SET
LongMonthNames='Enero;Febrero;Marzo;Abril;Mayo;Junio;Julio;Agosto;Septiembre;OctubreNoviembre;
Diciembre';

Transactions:
```

```
LOAD
date,
Date(date,'MMMM') as monthname,
id,
amount
INLINE
[
date,id,amount
01/01/2022,1,1000
02/01/2022,2,2123
03/01/2022,3,4124
04/01/2022,4,2431
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie `sum(amount)` als Kennzahl und die folgenden Felder als Dimensionen hinzu:

- `date`
- `monthname`

Erstellen Sie die folgende Kennzahl:

```
=sum(amount)
```

Ergebnistabelle

date	monthname	sum(amount)
01/01/2022	Enero	1000.45
01/02/2022	Enero	2123.34
01/03/2022	Enero	4124.35
01/04/2022	Enero	2431.36
01/05/2022	Enero	4787.78
01/06/2022	Enero	2431.84
01/07/2022	Enero	2854.83
01/08/2022	Enero	3554.28
01/09/2022	Enero	3756.17
01/10/2022	Enero	3454.35

Im Ladeskript wird die Variable `LongMonthNames` geändert, um die Monate des Jahres auf Spanisch aufzulisten. Dann erstellen Sie ein Feld mit dem Namen `monthname`. Die Funktion `Date` wird mit dem Feld `date` als erstem Argument verwendet. Das zweite Argument in der Funktion ist die Formatierung `MMMM`.

Anhand dieser Formatierung Qlik Sense werden die Werte aus dem ersten Argument in den entsprechenden vollständigen Monatsnamen konvertiert, der in der Variablen `LongMonthNames` festgelegt ist. In der Ergebnistabelle zeigen die Feldwerte des erstellten Felds `monthname` den Monatsnamen auf Spanisch an.

MoneyDecimalSep

Das angegebene Dezimaltrennzeichen ersetzt das von den Regionaleinstellungen vorgegebene Dezimaltrennzeichen für Währungen.

Standardmäßig zeigt Qlik Sense Zahlen und Text in Tabellendiagrammen anders an. Zahlen sind rechtsbündig, Text ist linksbündig. Dadurch lassen sich Probleme bei der Konvertierung von Text in Zahlen leicht auffinden. Alle Tabellen auf dieser Seite, die Qlik Sense-Ergebnisse anzeigen, verwenden diese Formatierung.

Syntax:

`MoneyDecimalSep`

Qlik Sense-Anwendungen interpretieren Textfelder, die dieser Formatierung als Währungswerte entsprechen. Das Textfeld muss das Währungssymbol enthalten, das in der Systemvariablen `MoneyFormat` definiert ist. `MoneyDecimalSep` ist besonders nützlich, wenn Datenquellen mit mehreren unterschiedlichen regionalen Einstellungen verarbeitet werden.

Das folgende Beispiel zeigt eine mögliche Nutzung der Systemvariablen `MoneyDecimalSep`

```
Set MoneyDecimalSep='.';
```

Diese Funktion wird oft zusammen mit den folgenden Funktionen verwendet:

Verwandte Funktionen

Funktion	Interaktion
`MoneyFormat`	In Fällen mit Textfeldinterpretation wird das Symbol `MoneyFormat` als Teil der Interpretation verwendet. Für das Zahlenformat wird das `MoneyFormat`-Format in Qlik Sense-Diagrammobjekten verwendet.
`MoneyThousandSep`	In Fällen mit Textfeldinterpretation muss auch die `MoneyThousandSep` beachtet werden.

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für

Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Beispiel 1 – Notation MoneyDecimalSep-Punkt (.)

Ladeskript und Ergebnisse

Überblick

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Einen Datensatz mit Datumsangaben, die in eine Tabelle mit dem Namen `Transactions` geladen werden.
- Es werden Daten bereitgestellt, deren Währungsfeld im Textformat mit einem Punkt „.“ als Dezimaltrennzeichen angegeben ist. Jedem Datensatz wird auch ein „$“ ´-Symbol vorangestellt, außer dem letzten Datensatz, dem ein „£“-Symbol vorangestellt wird.

Beachten Sie, dass die Systemvariable `MoneyFormat` das Dollarzeichen „$“ als Standardwährung definiert.

Ladeskript

```
SET MoneyThousandSep=',';
SET MoneyDecimalSep='.';
SET MoneyFormat='$###0.00;-$###0.00';

Transactions:
Load
date,
id,
amount
Inline
[
date,id,amount
01/01/2022,1,'$14.41'
01/02/2022,2,'$2,814.32'
01/03/2022,3,'$249.36'
01/04/2022,4,'$24.37'
01/05/2022,5,'$7.54'
01/06/2022,6,'$243.63'
01/07/2022,7,'$545.36'
01/08/2022,8,'$3.55'
01/09/2022,9,'$3.436'
01/10/2022,10,'£345.66'
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu:`amount`.

Fügen Sie die folgenden Kennzahlen hinzu:

- `isNum(amount)`
- `sum(amount)`

Prüfen Sie die Ergebnisse unten, die die korrekte Interpretation nur für alle Dollarwerte „$“ zeigt.

Ergebnistabelle

amount	=isNum(amount)	=Sum(amount)
Summen	0	$3905.98
£345.66	0	$0.00
$3.436	-1	$3.44
$3.55	-1	$3.55
$7.54	-1	$7.54
$14.41	-1	$14.41
$24.37	-1	$24.37
243.63	-1	$243.63
$249.36	-1	$249.36
$545.36	-1	$545.36
$2,814.32	-1	$2814.32

Die Ergebnistabelle oben zeigt, dass das Feld `amount` für alle Werte mit dem Dollar-Präfix ($) korrekt interpretiert wurde, während `amount` mit dem Pfund-Präfix (£) nicht in einen Währungswert konvertiert wurde.

Beispiel 2 – Notion MoneyDecimalSep-Komma (,)

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Datensatz, der in eine Tabelle namens `Transactions` geladen wird
- Es werden Daten bereitgestellt, deren Währungsfeld im Textformat mit einem Komma „,“ als Dezimaltrennzeichen angegeben ist. Jedem Datensatz wird auch ein „$“-Symbol vorangestellt, außer dem letzten Datensatz, der irrtümlicherweise das Punkt-Dezimaltrennzeichen (.) verwendet.

Beachten Sie, dass die Systemvariable `MoneyFormat` das Dollarzeichen „$“ als Standardwährung definiert.

Ladeskript

```
SET MoneyThousandSep='.';
SET MoneyDecimalSep=',';
SET MoneyFormat='$###0.00;-$###0.00';

Transactions:
Load
date,
id,
amount
Inline
[
date,id,amount
01/01/2022,1,'$14,41'
01/02/2022,2,'$2.814,32'
01/03/2022,3,'$249,36'
01/04/2022,4,'$24,37'
01/05/2022,5,'$7,54'
01/06/2022,6,'$243,63'
01/07/2022,7,'$545,36'
01/08/2022,8,'$3,55'
01/09/2022,9,'$3,436'
01/10/2022,10,'$345.66'
];
```

Ergebnisse

Absatztext für Ergebnisse.

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu:`amount`.

Fügen Sie die folgenden Kennzahlen hinzu:

- `isNum(amount)`
- `sum(amount)`

Prüfen Sie die Ergebnisse unten, die zeigen, dass alle Werte korrekt interpretiert wurden, außer dem Betrag, für den als Dezimaltrennzeichen die Punkt-Schreibweise (.) verwendet wird. In diesem Fall hätte stattdessen ein Komma verwendet werden müssen.

Ergebnistabelle

amount	=isNum(amount)	=Sum(amount)
Summen	0	$3905.98
$345.66	0	$0.00
$3,436	-1	$3.44
$3,55	-1	$3.55
$7,54	-1	$7.54
$14,41	-1	$14.41
$24,37	-1	$24.37
$243,63	-1	$243.63
$249,36	-1	$249.36
$545,36	-1	$545.36
$2.814,32	-1	$2814.32

MoneyFormat

Diese Systemvariable definiert das Formatmuster, das von Qlik für die automatische Übersetzung von Text in Zahlen verwendet wird, wobei der Zahl ein Währungssymbol vorangestellt wird. Sie definiert, wie Kennzahlen, deren Zahlenformateigenschaften auf „Money“ festgelegt sind, in Diagrammobjekten angezeigt werden.

Das als Teil des Formatmusters in der Systemvariablen `MoneyFormat` definierte Symbol ersetzt das von den Regionaleinstellungen vorgegebene Währungssymbol.

Standardmäßig zeigt Qlik Sense Zahlen und Text in Tabellendiagrammen anders an. Zahlen sind rechtsbündig, Text ist linksbündig. Dadurch lassen sich Probleme bei der Konvertierung von Text in Zahlen leicht auffinden. Alle Tabellen auf dieser Seite, die Qlik Sense-Ergebnisse anzeigen, verwenden diese Formatierung.

Syntax:

`MoneyFormat`

```
Set MoneyFormat='$ #,##0.00; ($ #,##0.00)';
```

Diese Formatierung wird in Diagrammobjekten angezeigt, wenn die Eigenschaft `Number Formatting` eines numerischen Felds auf `Money` festgelegt wird. Wenn außerdem numerische Textfelder von Qlik Sense interpretiert werden und das Währungssymbol des Textfelds dem in der Variablen `MoneyFormat` definierten Symbol entspricht, interpretiert Qlik Sense dieses Feld als Währungswert.

Diese Funktion wird oft zusammen mit den folgenden Funktionen verwendet:

Verwandte Funktionen

Funktion	Interaktion
MoneyDecimalSep (page 245)	Für das Zahlenformat wird `MoneyDecimalSep` in der Feldformatierung von Objekten verwendet.
MoneyThousandSep (page 253)	Für das Zahlenformat wird `MoneyThousandSep` in der Feldformatierung von Objekten verwendet.

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Beispiel 1 – MoneyFormat

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript enthält einen Datensatz, der in eine Tabelle mit dem Namen `Transactions` geladen wird. Die Standardvariablendefinition `MoneyFormat` wird verwendet.

Ladeskript

```
SET MoneyThousandSep=',';
SET MoneyDecimalSep='.';
SET MoneyFormat='$###0.00;-$###0.00';

Transactions:
Load
date,
id,
amount
Inline
[
date,id,amount
01/01/2022,1,$10000000441
```

```
01/02/2022,2,$21237492432
01/03/2022,3,$249475336
01/04/2022,4,$24313369837
01/05/2022,5,$7873578754
01/06/2022,6,$24313884663
01/07/2022,7,$545883436
01/08/2022,8,$35545828255
01/09/2022,9,$37565817436
01/10/2022,10,$3454343566
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `amount`

Fügen Sie folgende Kennzahl hinzu:

```
=Sum(amount)
```

Wählen Sie unter **Zahlenformat** die Option **Währung** aus, um `Sum(amount)` als Währungswert zu konfigurieren.

Ergebnistabelle

date	Betrag	=Sum(amount)
Summen		$165099674156.00
01/01/2022	$10000000441	$10000000441.00
01/02/2022	$21237492432	$21237492432.00
01/03/2022	$249475336	$249475336.00
01/04/2022	$24313369837	$24313369837.00
01/05/2022	$7873578754	$7873578754.00
01/06/2022	$24313884663	$24313884663.00
01/07/2022	$545883436	$545883436.00
01/08/2022	$35545828255	$35545828255.00
01/09/2022	$37565817436	$37565817436.00
01/10/2022	$3454343566	$3454343566.00

Die Standarddefinition `MoneyFormat` wird verwendet. Dies sieht wie folgt aus: `$###0.00;-$###0.00`. In der Ergebnistabelle zeigt das Format des Felds `amount` das Währungssymbol und das Dezimaltrennzeichen sowie die eingeschlossenen Dezimalstellen.

Beispiel 2 – „MoneyFormat“ mit Tausendertrennzeichen und gemischten Eingabeformaten

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz mit gemischten Eingabeformaten wird in eine Tabelle namens `Transactions` geladen und enthält Tausender- und Dezimaltrennzeichen.
- Die Definition von `MoneyFormat` wird geändert, um ein Komma als Tausendertrennzeichen einzuschließen.
- In einer der Datenzeilen wurden irrtümlich Kommas als Tausendertrennzeichen an den falschen Stellen gesetzt. Beachten Sie, wie dieser Betrag als Text zurückbleibt und nicht als Zahl interpretiert werden kann.

Ladeskript

```
SET MoneyThousandSep=',';
SET MoneyDecimalSep='.';
SET MoneyFormat = '$#,##0.00;-$#,##0.00';

Transactions:
Load
date,
id,
amount
Inline
[
date,id,amount
01/01/2022,1,'$10,000,000,441.45'
01/02/2022,2,'$212,3749,24,32.23'
01/03/2022,3,$249475336.45
01/04/2022,4,$24,313,369,837
01/05/2022,5,$7873578754
01/06/2022,6,$24313884663
01/07/2022,7,$545883436
01/08/2022,8,$35545828255
01/09/2022,9,$37565817436
01/10/2022,10,$3454343566
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- date
- amount

Fügen Sie folgende Kennzahl hinzu:

```
=Sum(amount)
```

Wählen Sie unter **Zahlenformat** die Option **Währung** aus, um Sum(amount) als Währungswert zu konfigurieren.

Ergebnistabelle

date	Betrag	=Sum(amount)
Summen		$119,548,811,911.90
01/01/2022	$10,000,000,441.45	$10,000,000,441.45
01/02/2022	$212,3749,24,32.23	$0.00
01/03/2022	$249475336.45	$249,475,336.45
01/04/2022	$24	$24.00
01/05/2022	$7873578754	$7,873,578,754.00
01/06/2022	$24313884663	$24,313,884,663.00
01/07/2022	$545883436	$545,883,436.00
01/08/2022	$35545828255	$35,545,828,255.00
01/09/2022	$37565817436	$37,565,817,436.00
01/10/2022	$3454343566	$3,454,343,566.00

Beim Start des Skripts wird die Systemvariable MoneyFormat geändert, um ein Komma als Tausendertrennzeichen einzuschließen. In der Tabelle Qlik Sense ist zu sehen, dass die Formatierung dieses Trennzeichen umfasst. Zudem wurde die Zeile mit dem falschen Trennzeichen nicht korrekt interpretiert und bleibt als Text zurück. Aus diesem Grund ist sie nicht in der Summe der Beträge enthalten.

MoneyThousandSep

Das angegebene Tausendertrennzeichen ersetzt das von den Regionaleinstellungen vorgegebene Tausendertrennzeichen für Währungen.

Standardmäßig zeigt Qlik Sense Zahlen und Text in Tabellendiagrammen anders an. Zahlen sind rechtsbündig, Text ist linksbündig. Dadurch lassen sich Probleme bei der Konvertierung von Text in Zahlen leicht auffinden. Alle Tabellen auf dieser Seite, die Qlik Sense-Ergebnisse anzeigen, verwenden diese Formatierung.

Syntax:

```
MoneyThousandSep
```

Qlik Sense-Anwendungen interpretieren Textfelder, die dieser Formatierung als Währungswerte entsprechen. Das Textfeld muss das Währungssymbol enthalten, das in der Systemvariablen `MoneyFormat` definiert ist. `MoneyThousandSep` ist besonders nützlich, wenn Datenquellen mit mehreren unterschiedlichen regionalen Einstellungen verarbeitet werden.

Das folgende Beispiel zeigt eine mögliche Nutzung der Systemvariablen `MoneyThousandSep`

```
Set MoneyDecimalSep=',';
```

Diese Funktion wird oft zusammen mit den folgenden Funktionen verwendet:

Verwandte Funktionen

Funktion	Interaktion
`MoneyFormat`	In Fällen mit Textfeldinterpretation wird das Symbol `MoneyFormat` als Teil der Interpretation verwendet. Für das Zahlenformat wird das `MoneyFormat`-Format von Qlik Sense in Diagrammobjekten verwendet.
`MoneyDecimalSep`	In Fällen mit Textfeldinterpretation muss auch die `MoneyDecimalSep` beachtet werden.

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Beispiel 1 – Notation MoneyThousandSep-Komma (,)

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Datensatz, der in eine Tabelle namens `Transactions` geladen wird
- Es werden Daten bereitgestellt, deren Währungsfeld im Textformat ein Komma „,“ als Tausendertrennzeichen aufweist. Jedem Datensatz wird auch ein „$“-Symbol vorangestellt.

Beachten Sie, dass die Systemvariable `MoneyFormat` das Dollarzeichen „$“ als Standardwährung definiert.

Ladeskript

```
SET MoneyThousandSep=',';
SET MoneyDecimalSep='.';
SET MoneyFormat='$###0.00;-$###0.00';

Transactions:
Load
date,
id,
amount
Inline
[
date,id,amount
01/01/2022,1,'$10,000,000,441'
01/02/2022,2,'$21,237,492,432'
01/03/2022,3,'$249,475,336'
01/04/2022,4,'$24,313,369,837'
01/05/2022,5,'$7,873,578,754'
01/06/2022,6,'$24,313,884,663'
01/07/2022,7,'$545,883,436'
01/08/2022,8,'$35,545,828,255'
01/09/2022,9,'$37,565,817,436'
01/10/2022,10,'$3.454.343.566'
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu:`amount`.

Fügen Sie die folgenden Kennzahlen hinzu:

- `isNum(amount)`
- `sum(amount)`

Sehen Sie unten die Ergebnisse ein. Die Tabelle zeigt die korrekte Interpretation aller Werte, die die Komma-Notation (,) als Tausendertrennzeichen verwenden.

Das Feld `amount` wurde für alle Werte korrekt interpretiert, mit Ausnahme eines Wertes, bei dem ein Punkt (.) als Tausendertrennzeichen verwendet wurde.

Ergebnistabelle

amount	=isNum(amount)	=Sum(amount)
Summen	0	$161645330590.00
$3.454.343.566	0	$0.00
$249,475,336	-1	$249475336.00
$545,883,436	-1	$545883436.00
$7,873,578,754	-1	$7873578754.00
$10,000,000,441	-1	$10000000441.00
$21,237,492,432	-1	$21237492432.00
$24,313,369,837	-1	$24313369837.00
$24,33,884,663	-1	$24313884663.00
$35,545,828,255	-1	$35545828255.00
$37,565,817,436	-1	$37565817436.00

Beispiel 2 – Notation MoneyThousandSep-Punkt (.)

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Datensatz, der in eine Tabelle namens `Transactions` geladen wird
- Es werden Daten bereitgestellt, deren Währungsfeld im Textformat einen Punkt „.“ als Tausendertrennzeichen aufweist. Jedem Datensatz wird auch ein „$“-Symbol vorangestellt.

Beachten Sie, dass die Systemvariable `MoneyFormat` das Dollarzeichen „$“ als Standardwährung definiert.

Ladeskript

```
SET MoneyThousandSep='.';
SET MoneyDecimalSep=',';
SET MoneyFormat='$###0.00;-$###0.00';

Transactions:
Load
date,
id,
amount
Inline
[
date,id,amount
```

```
01/01/2022,1,'$10.000.000.441'
01/02/2022,2,'$21.237.492.432'
01/03/2022,3,'$249.475.336'
01/04/2022,4,'$24.313.369.837'
01/05/2022,5,'$7.873.578.754'
01/06/2022,6,'$24.313.884.663'
01/07/2022,7,'$545.883.436'
01/08/2022,8,'$35.545.828.255'
01/09/2022,9,'$37.565.817.436'
01/10/2022,10,'$3,454,343,566'
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu:`amount`.

Fügen Sie die folgenden Kennzahlen hinzu:

- `isNum(amount)`
- `sum(amount)`

In den Ergebnissen unten ist zu sehen, dass alle Werte korrekt interpretiert wurden, die die Punkt-Notation (.) als Tausendertrennzeichen verwenden.

Das Feld `amount` wurde für alle Werte korrekt interpretiert, mit Ausnahme eines Wertes, bei dem ein Komma (,) als Tausendertrennzeichen verwendet wurde.

Ergebnistabelle

amount	=isNum(amount)	=Sum(amount)
Summen	0	$161645330590.00
$3,545,343,566	0	$0.00
$249.475.336	-1	$249475336.00
$545.883.436	-1	545883436.00
$7.873.578.754	-1	$7873578754.00
$10.000.000.441	-1	$10000000441.00
$21.237.492.432	-1	$21237492432.00
$24.313.884.663	-1	$24313884663.00
$24.313.884.663	-1	$24313884663.00
$35.545.828.255	-1	$35545828255.00
$37.565.817.436	-1	$37565817436.00

MonthNames

Das angegebene Format ersetzt das von den Regionaleinstellungen vorgegebene Format für Monatsnamen.

Syntax:

MonthNames

Wenn die Variable geändert wird, ist ein ; erforderlich, um die einzelnen Werte zu trennen.

Funktionsbeispiele

Beispiel	Ergebnisse
`Set MonthNames='Jan;Feb;Mar;Apr;May;Jun;Jul;Aug;Sep;Oct;Nov;Dec';`	Diese Verwendung der Funktion `MonthNames` definiert die Namen von Monaten auf Englisch und in ihrer abgekürzten Form.
`Set MonthNames='Enero;Feb;Marzo;Abr;Mayo;Jun;Jul;Agosto;Set;Oct;Nov;Dic';`	Diese Verwendung der Funktion `MonthNames` definiert die Namen von Monaten auf Spanisch und in ihrer abgekürzten Form.

Die Funktion `MonthNames` kann in Kombination mit den folgenden Funktionen verwendet werden:

Verwandte Funktionen

Funktion	Interaktion
month (page 936)	Skriptfunktion zum Zurückgeben von Werten, die in `MonthNames` als Feldwerte definiert sind.
Date (page 1258)	Skriptfunktion zum Zurückgeben von Werten, die in `MonthNames` als Feldwerte basierend auf dem bereitgestellten Formatierungsargument definiert sind.
LongMonthNames (page 241)	Langformwerte von `MonthNames`.

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben.

Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Beispiel 1 – Systemvariablenstandard

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz mit Datumsangaben, die in eine Tabelle mit dem Namen `Transactions` geladen werden.
- Ein Feld `date`.
- Die Standarddefinition `MonthNames`.

Ladeskript

```
SET MonthNames='Jan;Feb;Mar;Apr;May;Jun;Jul;Aug;Sep;Oct;Nov;Dec';

Transactions:
LOAD
date,
Month(date) as monthname,
id,
amount
INLINE
[
date,id,amount
01/01/2022,1,1000.45
01/02/2022,2,2123.34
01/03/2022,3,4124.35
01/04/2022,4,2431.36
01/05/2022,5,4787.78
01/06/2022,6,2431.84
01/07/2022,7,2854.83
01/08/2022,8,3554.28
01/09/2022,9,3756.17
01/10/2022,10,3454.35
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `monthname`

Erstellen Sie die folgende Kennzahl:

```
=sum(amount)
```

Ergebnistabelle

date	monthname	sum(amount)
01/01/2022	Jan	1000.45
01/02/2022	Jan	2123.34
01/03/2022	Jan	4124.35
01/04/2022	Jan	2431.36
01/05/2022	Jan	4787.78
01/06/2022	Jan	2431.84
01/07/2022	Jan	2854.83
01/08/2022	Jan	3554.28
01/09/2022	Jan	3756.17
01/10/2022	Jan	3454.35

Die Standarddefinition `MonthNames` wird verwendet. Im Ladeskript wird die Funktion `Month` mit dem Feld `date` als bereitgestelltes Argument verwendet.

In der Ergebnistabelle zeigt die Ausgabe dieser Funktion `Month` die Monate des Jahres im Format der Definition `MonthNames` an.

Beispiel 2 – Systemvariable ändern

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Einen Datensatz mit Datumsangaben, die in eine Tabelle mit dem Namen `Transactions` geladen werden.

- Ein Feld date.
- Die Variable MonthNames wird geändert, um die abgekürzten Monate auf Spanisch zu verwenden.

Ladeskript

```
                          Set
MonthNames='Enero;Feb;Marzo;Abr;Mayo;Jun;Jul;Agosto;Set;Oct;Nov;Dic';

Transactions:
LOAD
date,
month(date) as month,
id,
amount
INLINE
[
date,id,amount
01/01/2022,1,1000
02/01/2022,2,2123
03/01/2022,3,4124
04/01/2022,4,2431
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- date
- monthname

Erstellen Sie die folgende Kennzahl:

```
=sum(amount)
```

Ergebnistabelle

date	monthname	sum(amount)
01/01/2022	Enero	1000.45
01/02/2022	Enero	2123.34
01/03/2022	Enero	4124.35
01/04/2022	Enero	2431.36
01/05/2022	Enero	4787.78
01/06/2022	Enero	2431.84
01/07/2022	Enero	2854.83
01/08/2022	Enero	3554.28
01/09/2022	Enero	3756.17
01/10/2022	Enero	3454.35

Im Ladeskript wird zuerst die Variable MonthNames geändert, um die Monate des Jahres auf Spanisch aufzulisten. Die Funktion Month wird mit dem Feld date als bereitgestelltem Argument verwendet.

In der Ergebnistabelle zeigt die Ausgabe dieser Funktion Month die Monate des Jahres im Format der Definition MonthNames an.

Beachten Sie: Wenn die Sprache für die Variable MonthNames wie in diesem Beispiel geändert wird, enthält die Variable LongMonthNames nach wie vor die Monate des Jahres auf Englisch. Die Variable LongMonthNames muss ebenfalls geändert werden, wenn beide Variablen in der Anwendung verwendet werden.

Beispiel 3 – Datumsfunktion

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Einen Datensatz mit Datumsangaben, die in eine Tabelle mit dem Namen Transactions geladen werden.
- Ein Feld date.
- Die Standarddefinition MonthNames.

Ladeskript

```
SET MonthNames='Jan;Feb;Mar;Apr;May;Jun;Jul;Aug;Sep;Oct;Nov;Dec';

Transactions:
LOAD
date,
Month(date, 'MMM') as monthname,
id,
amount
INLINE
[
date,id,amount
01/01/2022,1,1000.45
01/02/2022,2,2123.34
01/03/2022,3,4124.35
01/04/2022,4,2431.36
01/05/2022,5,4787.78
01/06/2022,6,2431.84
01/07/2022,7,2854.83
01/08/2022,8,3554.28
01/09/2022,9,3756.17
01/10/2022,10,3454.35
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `monthname`

Erstellen Sie die folgende Kennzahl:

```
=sum(amount)
```

Ergebnistabelle

date	monthname	sum(amount)
01/01/2022	Jan	1000.45
01/02/2022	Jan	2123.34
01/03/2022	Jan	4124.35
01/04/2022	Jan	2431.36
01/05/2022	Jan	4787.78
01/06/2022	Jan	2431.84
01/07/2022	Jan	2854.83
01/08/2022	Jan	3554.28
01/09/2022	Jan	3756.17
01/10/2022	Jan	3454.35

Die Standarddefinition `MonthNames` wird verwendet. Im Ladeskript wird die Funktion `Date` mit dem Feld `date` als erstes Argument verwendet. Das zweite Argument ist `MMM`.

Anhand dieser Formatierung Qlik Sense werden die Werte aus dem ersten Argument in den entsprechenden Monatsnamen konvertiert, der in der Variablen `MonthNames` festgelegt ist. In der Ergebnistabelle wird dies in den Feldwerten des neu erstellten Felds `month` gezeigt.

NumericalAbbreviation

Mit der numerischen Abkürzung wird festgelegt, welche Abkürzung für Größenordnungspräfixe von Zahlen verwendet werden, z. B. M für Mega oder eine Million (10^6) und µ für Mikro (10^{-6}).

Syntax:

`NumericalAbbreviation`

Sie legen die Variable `NumericalAbbreviation` auf einen String fest, der eine Liste von Abkürzungsdefinitionspaaren enthält, die durch Strichpunkte voneinander getrennt sind. Jedes Abkürzungsdefinitionspaar muss die Größenordnung (den Exponenten in der Dezimalbasis) sowie die durch einen Doppelpunkt abgetrennte Abkürzung enthalten, z. B. `6:M` für eine Million.

Die Standardeinstellung ist `'3:k;6:M;9:G;12:T;15:P;18:E;21:Z;24:Y;-3:m;-6:µ;-9:n;-12:p;-15:f;-18:a;-21:z;-24:y'`.

Beispiele:

Mit dieser Einstellung wird das Präfix für Tausend zu t und für eine Milliarde zu B geändert. Das ist nützlich für Finanzanwendungen in denen Abkürzungen wie t$ und M$ zu erwarten sind.

```
Set NumericalAbbreviation='3:t;6:M;9:B;12:T;15:P;18:E;21:Z;24:Y;-3:m;-6:µ;-9:n;-12:p;-15:f;-18:a;-21:z;-24:y';
```

ReferenceDay

Die Einstellung definiert, welcher Tag im Januar als Referenztag zum Definieren von Woche 1 festgelegt wird. Die Einstellung legt also fest, wie viele Tage in Woche 1 ein Datum im Januar aufweisen müssen.

Syntax:

```
ReferenceDay
```

`ReferenceDay` legt fest, wie viele Tage in der ersten Woche des Jahres enthalten sind. `ReferenceDay` kann auf einen beliebigen Wert zwischen `1` und `7` festgelegt werden. Jeder Wert außerhalb des Bereichs `1-7` wird als Wochenmitte (`4`) interpretiert, was dem Festlegen von `ReferenceDay` auf `4` entspricht.

Wenn Sie keinen Wert für die Einstellung `ReferenceDay` festlegen, zeigt der Standardwert `ReferenceDay=0`, was als Wochenmitte (`4`) ausgelegt wird, wie in der Wertetabelle `ReferenceDay` unten gezeigt.

Die Funktion `ReferenceDay` wird oft in Kombination mit den folgenden Funktionen verwendet:

Verwandte Funktionen

Variable	Interaktion
BrokenWeeks (page 219)	Wenn die Qlik Sense-App mit vollständigen Wochen arbeitet, wird die Einstellung der Variablen `ReferenceDay` erzwungen. Wenn jedoch unvollständige Wochen verwendet werden, beginnt Woche 1 am 1. Januar und endet entsprechend der Variableneinstellung `FirstWeekDay`, wobei das Kennzeichen `ReferenceDay` ignoriert wird.
FirstWeekDay (page 233)	Ganzzahlwert, der definiert, welcher Tag als erster Tag der Woche verwendet wird.

In Qlik Sense können die folgenden Werte für `ReferenceDay` festgelegt werden:

Werte für „ReferenceDay“

Wert	Referenztag
0 (Standard)	4. Januar
1	1. Januar
2	January 2

Wert	Referenztag
3	3. Januar
4	4. Januar
5	5. Januar
6	6. Januar
7	7. Januar

Im folgenden Beispiel definiert `ReferenceDay = 3` den 3. Januar als Referenztag:

```
SET ReferenceDay=3; //(set January 3 as the reference day)
```

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Beispiele:

Wenn Sie ISO-Einstellungen für Wochen und Wochennummern verwenden möchten, müssen Sie Folgendes in das Skript einschließen:

```
Set FirstWeekDay=0;
Set BrokenWeeks=0;
Set ReferenceDay=4;    // Jan 4th is always in week 1
```

Wenn Sie US-Einstellungen verwenden möchten, müssen Sie Folgendes in das Skript einschließen:

```
Set FirstWeekDay=6;
Set BrokenWeeks=1;
Set ReferenceDay=1;     // Jan 1st is always in week 1
```

Beispiel 1 – Ladeskript mit Verwendung des Standardwerts; ReferenceDay=0

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Die Variable `ReferenceDay`, die auf `0` festgelegt ist.
- Die Variable `BrokenWeeks`, die auf `0` festgelegt ist, wodurch erzwungen wird, dass die App vollständige Wochen verwendet.
- Ein Datensatz mit Datumsangaben von Ende 2019 bis Anfang 2020.

Ladeskript

```
SET BrokenWeeks = 0;
SET ReferenceDay = 0;

Sales:
LOAD
date,
sales,
week(date) as week,
weekday(date) as weekday
Inline [
date,sales
12/27/2019,5000
12/28/2019,6000
12/29/2019,7000
12/30/2019,4000
12/31/2019,3000
01/01/2020,6000
01/02/2020,3000
01/03/2020,6000
01/04/2020,8000
01/05/2020,5000
01/06/2020,7000
01/07/2020,3000
01/08/2020,5000
01/09/2020,9000
01/10/2020,5000
01/11/2020,7000
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `week`
- `weekday`

Ergebnistabelle

date	Woche	weekday
12/27/2019	52	Fr

date	Woche	weekday
12/28/2019	52	Sa
12/29/2019	1	So
12/30/2019	1	Mo
12/31/2019	1	Di
01/01/2020	1	Mi
01/02/2020	1	Do
01/03/2020	1	Fr
01/04/2020	1	Sa
01/05/2020	2	So
01/06/2020	2	Mo
01/07/2020	2	Di
01/08/2020	2	Mi
01/09/2020	2	Do
01/10/2020	2	Fr
01/11/2020	2	Sa

Woche 52 endet am Samstag, den 28. Dezember. Da `ReferenceDay` erfordert, dass der 4. Januar in Woche 1 eingeschlossen wird, beginnt Woche 1 am 29. Dezember und endet am Samstag, den 4. Januar.

Beispiel – Variable „ReferenceDay“ auf 5 festgelegt

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Die Variable `ReferenceDay`, die auf `5` festgelegt ist.
- Die Variable `BrokenWeeks`, die auf `0` festgelegt ist, wodurch erzwungen wird, dass die App vollständige Wochen verwendet.
- Ein Datensatz mit Datumsangaben von Ende 2019 bis Anfang 2020.

Ladeskript

```
SET BrokenWeeks = 0;
SET ReferenceDay = 5;

Sales:
LOAD
date,
```

```
sales,
week(date) as week,
weekday(date) as weekday
Inline [
date,sales
12/27/2019,5000
12/28/2019,6000
12/29/2019,7000
12/30/2019,4000
12/31/2019,3000
01/01/2020,6000
01/02/2020,3000
01/03/2020,6000
01/04/2020,8000
01/05/2020,5000
01/06/2020,7000
01/07/2020,3000
01/08/2020,5000
01/09/2020,9000
01/10/2020,5000
01/11/2020,7000
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- date
- week
- weekday

Ergebnistabelle

date	**Woche**	**weekday**
12/27/2019	52	Fr
12/28/2019	52	Sa
12/29/2019	53	So
12/30/2019	53	Mo
12/31/2019	53	Di
01/01/2020	53	Mi
01/02/2020	53	Do
01/03/2020	53	Fr
01/04/2020	53	Sa
01/05/2020	1	So
01/06/2020	1	Mo

date	Woche	weekday
01/07/2020	1	Di
01/08/2020	1	Mi
01/09/2020	1	Do
01/10/2020	1	Fr
01/11/2020	1	Sa

Woche 52 endet am Samstag, den 28. Dezember. Die Variable `Brokenweeks` erzwingt, dass die App vollständige Wochen verwendet. Der Referenztagwert von `5` erfordert, dass der 5. Januar in Woche 1 liegt.

Er liegt aber acht Tage nach dem Ende von Woche 52 des Vorjahres. Daher beginnt Woche 53 am 29. Dezember und endet am 4. Januar. Woche 1 beginnt am Sonntag, den 5. Januar.

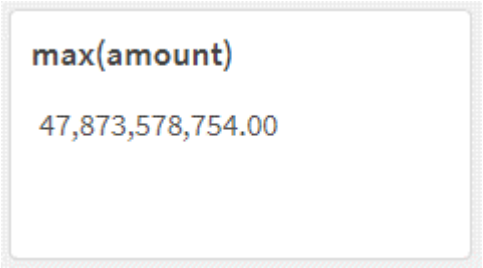
ThousandSep

Das angegebene Tausendertrennzeichen ersetzt das vom Betriebssystem vorgegebene Tausendertrennzeichen (Regionaleinstellungen).

Syntax:

`ThousandSep`

Qlik Sense-Objekt, das die Variable `ThousandSep` (mit Tausendertrennzeichen) verwendet



Qlik Sense-Apps interpretieren Textfelder in diesem Format als Zahlen. Dieses Format wird in Diagrammobjekten angezeigt, wenn die Eigenschaft **Zahlenformat** eines numerischen Felds auf **Zahl** festgelegt ist.

`ThousandSep` ist nützlich bei der Handhabung von Datenquellen, die mehreren regionalen Einstellungen entstammen.

> *Wenn die Variable `ThousandSep` geändert wird, nachdem bereits Objekte erstellt und in der Anwendung formatiert wurden, muss der Benutzer jedes relevante Feld erneut formatieren, indem er die Auswahl der Eigenschaft **Zahlenformat** aufhebt und dann erneut **Zahl** auswählt.*

Die folgenden Beispiele zeigen eine mögliche Nutzung der Systemvariablen `ThousandSep`:

```
Set ThousandSep=','; //(for example, seven billion will be displayed as: 7,000,000,000)

Set ThousandSep=' '; //(for example, seven billion will be displayed as: 7 000 000 000)
```

Die folgenden Themen können Sie bei der Arbeit mit dieser Funktion unterstützen:

Verwandte Themen

Thema	Beschreibung
DecimalSep (page 231)	In Fällen der Textfeldinterpretation müssen auch die Einstellungen für das Dezimaltrennzeichen berücksichtigt werden, die durch diese Funktion bereitgestellt werden. Für das Zahlenformat wird bei Bedarf **DecimalSep** von Qlik Sense verwendet.

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Beispiel 1 – Standardsystemvariablen

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Datensatz, der in eine Tabelle namens `Transactions` geladen wird
- Verwendung der Standardvariablendefinition `ThousandSep`

Ladeskript

```
Transactions:
Load
date,
id,
amount
Inline
[
date,id,amount
01/01/2022,1,10000000441
01/02/2022,2,21237492432
```

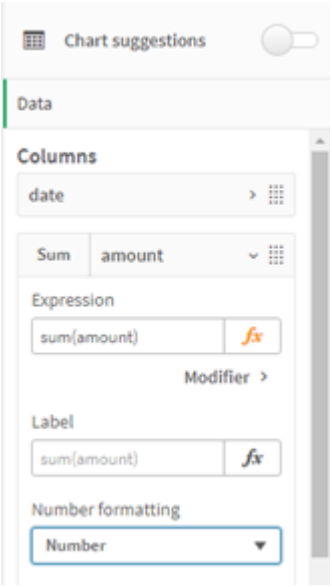
```
01/03/2022,3,41249475336
01/04/2022,4,24313369837
01/05/2022,5,47873578754
01/06/2022,6,24313884663
01/07/2022,7,28545883436
01/08/2022,8,35545828255
01/09/2022,9,37565817436
01/10/2022,10,3454343566
];
```

Ergebnisse

Gehen Sie folgendermaßen vor:

1. Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu:`date`.
2. Fügen Sie die folgende Kennzahl hinzu:
   `=sum(amount)`
3. Wählen Sie im Eigenschaftsfenster unter **Daten** die Kennzahl aus.
4. Wählen Sie unter **Zahlenformat** die Option **Zahl** aus.

Anpassen des Zahlenformats für eine Diagrammkennzahl



Ergebnistabelle

date	=sum(amount)
01/01/2022	10,000,000,441.00
01/02/2022	21,237,492,432.00
01/03/2022	41,249,475,336.00

date	=sum(amount)
01/04/2022	24,313,369,837.00
01/05/2022	47,873,578,754.00
01/06/2022	24,313,884,663.00
01/07/2022	28,545,883,436.00
01/08/2022	35,545,828,255.00
01/09/2022	37,565,817,436.00
01/10/2022	3,454,343,566.00

In diesem Beispiel wird die Standarddefinition `ThousandSep` verwendet, die auf das Kommaformat (,) festgelegt ist. In der Ergebnistabelle wird im Format des Betragsfelds ein Komma zwischen Tausendergruppen angezeigt.

Beispiel 2 – Ändern der Systemvariablen

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Der gleiche Datensatz wie im ersten Beispiel wird in eine Tabelle namens `Transactions` geladen.
- Die Definition von `ThousandSep` wird am Beginn des Skripts zu einem *-Zeichen als Tausendertrennzeichen geändert. Dies ist ein extremes Beispiel, das nur verwendet wird, um die Funktionalität der Variablen zu demonstrieren.

Die in diesem Beispiel verwendete Änderung ist extrem und wird nicht üblicherweise verwendet. Hier wird sie gezeigt, um die Funktionalität der Variablen zu demonstrieren.

Ladeskript

```
SET ThousandSep='*';

Transactions:
Load
date,
id,
amount
Inline
[
date,id,amount
01/01/2022,1,10000000441
01/02/2022,2,21237492432
01/03/2022,3,41249475336
```

```
01/04/2022,4,24313369837
01/05/2022,5,47873578754
01/06/2022,6,24313884663
01/07/2022,7,28545883436
01/08/2022,8,35545828255
01/09/2022,9,37565817436
01/10/2022,10,3454343566
];
```

Ergebnisse

Gehen Sie folgendermaßen vor:

1. Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu:`date`.
2. Fügen Sie die folgende Kennzahl hinzu:
   `=sum(amount)`
3. Wählen Sie im Eigenschaftsfenster unter **Daten** die Kennzahl aus.
4. Wählen Sie unter **Zahlenformat** die Option **Benutzerdefiniert** aus.

Ergebnistabelle

date	=sum(amount)
01/01/2022	10*000*000*441.00
01/02/2022	21*237*492*432.00
01/03/2022	41*249*475*336.00
01/04/2022	24*313*369*837.00
01/05/2022	47*873*578*754.00
01/06/2022	24*313*884*663.00
01/07/2022	28*545*883*436.00
01/08/2022	35*545*828*255.00
01/09/2022	37*565*817*436.00
01/10/2022	3*454*343*566.00

Am Beginn des Skripts wird die Systemvariable `ThousandSep` zu „*“ geändert. In der Ergebnistabelle wird im Format des Betragsfelds ein „*“ zwischen Tausendergruppen angezeigt.

Beispiel 3 – Textinterpretation

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Datensatz, der in eine Tabelle namens `Transactions` geladen wird
- Daten, die ein numerisches Feld im Textformat aufweisen, wobei ein Komma als Tausendertrennzeichen verwendet wird
- Verwendung der Standardsystemvariable `ThousandSep`

Ladeskript

```
Transactions:
Load
date,
id,
amount
Inline
[
date,id,amount
01/01/2022,1,'10,000,000,441'
01/02/2022,2,'21,492,432'
01/03/2022,3,'4,249,475,336'
01/04/2022,4,'24,313,369,837'
01/05/2022,5,'4,873,578,754'
01/06/2022,6,'313,884,663'
01/07/2022,7,'2,545,883,436'
01/08/2022,8,'545,828,255'
01/09/2022,9,'37,565,817,436'
01/10/2022,10,'3,454,343,566'
];
```

Ergebnisse

Gehen Sie folgendermaßen vor:

1. Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu:`date`.
2. Fügen Sie die folgende Kennzahl hinzu:
   `=sum(amount)`
3. Wählen Sie im Eigenschaftsfenster unter **Daten** die Kennzahl aus.
4. Wählen Sie unter **Zahlenformat** die Option **Zahl** aus.
5. Fügen Sie die folgende Kennzahl hinzu, um auszuwerten, ob das Betrags-Feld ein numerischer Wert ist oder nicht:
   `=isnum(amount)`

Ergebnistabelle

date	=sum(amount)	=isnum(amount)
01/01/2022	10,000,000,441.00	-1
01/02/2022	21,492,432.00	-1

date	=sum(amount)	=isnum(amount)
01/03/2022	4,249,475,336.00	-1
01/04/2022	24,313,369,837.00	-1
01/05/2022	4,873,578,754.00	-1
01/06/2022	313,884,663.00	-1
01/07/2022	2,545,883,436.00	-1
01/08/2022	545,828,255.00	-1
01/09/2022	37,565,817,436.00	-1
01/10/2022	3*454*343*566.00	-1

Nachdem die Daten geladen wurden, ist zu sehen, dass Qlik Sense das Betragsfeld als numerischen Wert interpretiert hat, da die Daten der Variablen `ThousandSep` entsprechen. Dies wird durch die Funktion `isnum()` gezeigt, die jeden Eintrag als -1 bzw. `TRUE` auswertet.

In Qlik Sense wird der boolesche Wert „wahr“ durch -1 dargestellt, der Wert „falsch“ durch 0.

TimeFormat

Das angegebene Format für Uhrzeiten ersetzt das vom Betriebssystem vorgegebene Zeitformat (Regionaleinstellungen).

Syntax:

```
TimeFormat
```

Beispiel:

```
Set TimeFormat='hh:mm:ss';
```

TimestampFormat

Das angegebene Format für Zeitstempel (Datum und Uhrzeit) ersetzt das vom Betriebssystem vorgegebene (Regionaleinstellungen).

Syntax:

```
TimestampFormat
```

Beispiel:

In den folgenden Beispielen wird *1983-12-14T13:15:30Z* als Zeitstempeldaten verwendet, um die Ergebnisse der einzelnen **SET TimestampFormat**-Befehle zu zeigen. Das verwendete Datenformat ist **YYYYMMDD** und das Zeitformat ist **h:mm:ss TT**. Das Datumsformat wird im Befehl **SET DateFormat** und das Zeitformat im Befehl **SET TimeFormat** oben im Datenladeskript angegeben.

Results

Beispiel	Ergebnis
`SET TimestampFormat='YYYYMMDD';`	19831214
`SET TimestampFormat='M/D/YY hh:mm:ss[.fff]';`	12/14/83 13:15:30
`SET TimestampFormat='DD/MM/YYYY hh:mm:ss[.fff]';`	14/12/1983 13:15:30
`SET TimestampFormat='DD/MM/YYYY hh:mm:ss[.fff] TT';`	14/12/1983 1:15:30 PM
`SET TimestampFormat='YYYY-MM-DD hh:mm:ss[.fff] TT';`	1983-12-14 01:15:30

Beispiele: Ladeskript

Beispiel: Ladeskript
Im ersten Ladeskript wird *SET TimestampFormat='DD/MM/YYYY h:mm:ss[.fff] TT'* verwendet. Im zweiten Ladeskript wird das Zeitstempelformat in *SET TimestampFormat='MM/DD/YYYY hh:mm:ss[.fff]'* geändert. Die verschiedenen Ergebnisse zeigen, wie der Befehl **SET TimeFormat** zusammen mit verschiedenen Zeitdatenformaten funktioniert.

Die Tabelle unten zeigt den Datensatz, der in den folgenden Ladeskripts verwendet wird. Die zweite Spalte der Tabelle zeigt das Format jedes Zeitstempels im Datensatz. Die ersten fünf Zeitstempel folgen den Regeln von ISO 8601, der sechste jedoch nicht.

Datensatz

Tabelle mit den verwendeten Zeitdaten und dem Format für jeden Zeitstempel im Datensatz.

transaction_timestamp	time data format
2018-08-30	YYYY-MM-DD
20180830T193614.857	YYYYMMDDhhmmss.sss
20180830T193614.857+0200	YYYYMMDDhhmmss.sss±hhmm
2018-09-16T12:30-02:00	YYYY-MM-DDhh:mm±hh:mm
2018-09-16T13:15:30Z	YYYY-MM-DDhh:mmZ
9/30/18 19:36:14	M/D/YY hh:mm:ss

Erstellen Sie im **Dateneditor** einen neuen Abschnitt, fügen Sie dann das Beispielskript hinzu und führen Sie es aus. Fügen Sie einem Arbeitsblatt in Ihrer App dann mindestens diejenigen Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um das Ergebnis anzuzeigen.

Ladeskript

```
SET FirstWeekDay=0;
SET BrokenWeeks=1;
SET ReferenceDay=0;
SET DayNames='Mon;Tue;Wed;Thu;Fri;Sat;Sun';
SET LongDayNames='Monday;Tuesday;Wednesday;Thursday;Friday;Saturday;Sunday';
```

```
SET DateFormat='YYYYMMDD';
SET TimestampFormat='DD/MM/YYYY h:mm:ss[.fff] TT';

Transactions:
Load
*,
Timestamp(transaction_timestamp, 'YYYY-MM-DD hh:mm:ss[.fff]') as LogTimestamp
;

Load * Inline [
transaction_id, transaction_timestamp, transaction_amount, transaction_quantity, discount,
customer_id, size, color_code
3750, 2018-08-30, 12423.56, 23, 0,2038593, L, Red
3751, 20180830T193614.857, 5356.31, 6, 0.1, 203521, m, orange
3752, 20180830T193614.857+0200, 15.75, 1, 0.22, 5646471, S, blue
3753, 2018-09-16T12:30-02:00, 1251, 7, 0, 3036491, l, Black
3754, 2018-09-16T13:15:30Z, 21484.21, 1356, 75, 049681, xs, Red
3755, 9/30/18 19:36:14, -59.18, 2, 0.333333333333333, 2038593, M, Blue
];
```

Ergebnisse

Qlik Sense Tabelle mit den Ergebnissen der Variablen zur Interpretation TimestampFormat, die im Ladeskript verwendet wird. Der letzte Zeitstempel im Datensatz gibt kein korrektes Datum zurück.

transaction_id	transaction_timestamp	LogTimeStamp
3750	2018-08-30	2018-08-30 00:00:00
3751	20180830T193614.857	2018-08-30 19:36:14
3752	20180830T193614.857+0200	2018-08-30 17:36:14
3753	2018-09-16T12:30-02:00	2018-09-16 14:30:00
3754	2018-09-16T13:15:30Z	2018-09-16 13:15:30
3755	9/30/18 19:36:14	-

Das nächste Ladeskript verwendet den gleichen Datensatz. Es verwendet jedoch *SET TimestampFormat='MM/DD/YYYY hh:mm:ss[.fff]'*, um dem Nicht-ISO-8601-Format des sechsten Zeitstempels zu entsprechen.

Ersetzen Sie im **Dateneditor** das vorherige Beispielskript durch das Skript unten und führen Sie es aus. Fügen Sie einem Arbeitsblatt in Ihrer App dann mindestens diejenigen Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um das Ergebnis anzuzeigen.

Ladeskript

```
SET FirstWeekDay=0;
SET BrokenWeeks=1;
SET ReferenceDay=0;
SET DayNames='Mon;Tue;Wed;Thu;Fri;Sat;Sun';
SET LongDayNames='Monday;Tuesday;Wednesday;Thursday;Friday;Saturday;Sunday';
```

```
SET DateFormat='YYYYMMDD';
SET TimestampFormat='MM/DD/YYYY hh:mm:ss[.fff]';

Transactions:
Load
*,
Timestamp(transaction_timestamp, 'YYYY-MM-DD hh:mm:ss[.fff]') as LogTimestamp
;

Load * Inline [
transaction_id, transaction_timestamp, transaction_amount, transaction_quantity, discount,
customer_id, size, color_code
3750, 2018-08-30, 12423.56, 23, 0,2038593, L, Red
3751, 20180830T193614.857, 5356.31, 6, 0.1, 203521, m, orange
3752, 20180830T193614.857+0200, 15.75, 1, 0.22, 5646471, S, blue
3753, 2018-09-16T12:30-02:00, 1251, 7, 0, 3036491, l, Black
3754, 2018-09-16T13:15:30Z, 21484.21, 1356, 75, 049681, xs, Red
3755, 9/30/18 19:36:14, -59.18, 2, 0.333333333333333, 2038593, M, Blue
];
```

Ergebnisse

Qlik Sense Tabelle mit den Ergebnissen der Variablen zur Interpretation TimestampFormat, die im Ladeskript verwendet wird.

transaction_id	transaction_timestamp	LogTimeStamp
3750	2018-08-30	2018-08-30 00:00:00
3751	20180830T193614.857	2018-08-30 19:36:14
3752	20180830T193614.857+0200	2018-08-30 17:36:14
3753	2018-09-16T12:30-02:00	2018-09-16 14:30:00
3754	2018-09-16T13:15:30Z	2018-09-16 13:15:30
3755	9/30/18 19:36:14	2018-09-16 19:36:14

4.9 Direct Discovery-Variablen

Direct Discovery-Systemvariablen

DirectCacheSeconds

Sie können ein Caching-Limit zu den Direct Discovery-Abfrageergebnissen für Visualisierungen festsetzen. Bei Erreichen dieses Zeitlimits löscht Qlik Sense den Cache, wenn neue Direct Discovery-Abfragen erfolgen. Qlik Sense fragt die Quelldaten nach den Auswahlen ab und erstellt den Cache erneut für das festgelegte Zeitlimit. Das Ergebnis für jede Kombination von Auswahlen wird unabhängig gecacht. Das heißt, der Cache wird für jede Auswahl unabhängig aktualisiert. Also aktualisiert eine Auswahl den Cache nur für die ausgewählten Felder und eine zweite Auswahl aktualisiert den Cache für deren relevante Felder. Falls die zweite Auswahl Felder einschließt, die in der ersten Auswahl aktualisiert wurden, werden sie im Cache nicht erneut aktualisiert, wenn das Caching-Limit nicht erreicht wurde.

Der Direct Discovery-Cache bezieht sich nicht auf die **Tabellen**visualisierungen. Tabellenauswahlen fragen die Datenquelle jedes Mal ab.

Der Limit-Wert muss in Sekunden festgesetzt werden. Das standardmäßige Cache-Limit beträgt 1800 Sekunden (30 Minuten).

Der für **DirectCacheSeconds** verwendete Wert ist der Wert, der gleichzeitig mit der Ausführung des **DIRECT QUERY**-Befehls festgesetzt wird. Der Wert kann während der Laufzeit nicht geändert werden.

Beispiel:

```
SET DirectCacheSeconds=1800;
```

DirectConnectionMax
Mithilfe der Verbindungspooling-Funktion können Sie asynchrone, parallele Abrufe aus der Datenbank durchführen. Die Syntax des Ladeskripts zur Einrichtung der Pooling-Funktion sieht wie folgt aus:

```
SET DirectConnectionMax=10;
```
Die numerische Einstellung gibt die maximale Anzahl von Datenbankverbindungen an, die der Direct Discovery-Code verwenden soll, während ein Arbeitsblatt aktualisiert wird. Standard ist 1.

Diese Variable sollte mit Vorsicht angewendet werden. Sie auf höher als 1 zu setzen, verursacht bekanntlich Probleme beim Verbinden mit Microsoft SQL Server.

DirectUnicodeStrings
Direct Discovery kann die Auswahl erweiterter Unicode-Daten unterstützen, und zwar durch Verwendung des SQL-Standardformats für erweiterte Zeichenketten-Literale (N’<extended string>’), so wie dies für manche Datenbanken erforderlich ist (insbesondere SQL Server). Die Verwendung dieser Syntax kann für Direct Discovery mit der Skriptvariable **DirectUnicodeStrings** aktiviert werden.

Wenn für diese Variable "wahr" eingestellt wird, dann ist die Verwendung des ANSI-normweiten Zeichen-Markers “N” vor den String-Literalen aktiviert. Nicht alle Datenbanken unterstützen diesen Standard. Standardeinstellung ist "falsch".

DirectDistinctSupport
Bei Auswählen eines **DIMENSION**-Felds in einem Qlik Sense-Objekt wird eine Abfrage für die Quelldatenbank generiert. Wenn die Abfrage eine Gruppierung erfordert, verwendet Direct Discovery das Schlüsselwort **DISTINCT**, um nur eindeutige Werte auszuwählen. Einige Datenbanken erfordern jedoch das Schlüsselwort **GROUP BY**. Setzen Sie **DirectDistinctSupport** auf 'false', um **GROUP BY** anstelle von **DISTINCT** in Abfragen für eindeutige Werte zu generieren.

```
SET DirectDistinctSupport='false';
```
Wenn DirectDistinctSupport auf wahr gesetzt ist, wird **DISTINCT** verwendet. Wenn dies nicht gesetzt ist, muss das Standardverhalten **DISTINCT** verwenden.

DirectEnableSubquery
In Szenarien mit hoher Kardinalität und mehreren Tabellen können anstelle einer großen SQL-Bedingung mehrere Unterabfragen in der IN-Abfrage generiert werden. Dies wird aktiviert, indem **DirectEnableSubquery** auf 'true' gesetzt wird. Der Standardwert ist 'false'.

*Wenn **DirectEnableSubquery** aktiviert ist, können Sie keine Tabellen laden, die nicht im Direct Discovery-Modus sind.*

```
SET DirectEnableSubquery='true';
```

Teradata Query-Banding-Variablen

Teradata Query Banding ist eine Funktion, die Unternehmensanwendungen mit der zugrunde liegenden Teradata-Datenbank für bessere Buchhaltung, Priorisierung und Workload Management zusammenarbeiten lässt. Mithilfe von Query Banding können Sie Metadaten wie Login-Daten von Benutzern um eine Abfrage packen.

Zwei Variablen sind verfügbar, beide sind Strings, die evaluiert und an die Datenbank gesendet werden.

SQLSessionPrefix
Dieser String wird beim Erstellen einer Verbindung mit der Datenbank gesendet.

```
SET SQLSessionPrefix = 'SET QUERY_BAND = ' & Chr(39) & 'Who=' & OSuser() & ';' & Chr(39) & '
FOR SESSION;';
```
Wenn **OSuser()** zum Beispiel *WA\sbt* ergibt, wird dies als `SET QUERY_BAND = 'Who=WA\sbt;' FOR SESSION;` bewertet, was beim Erstellen der Verbindung an die Datenbank gesendet wird.

SQLQueryPrefix
Dieser String wird für jede einzelne Abfrage gesendet.

```
SET SQLSessionPrefix = 'SET QUERY_BAND = ' & Chr(39) & 'Who=' & OSuser() & ';' & Chr(39) & '
FOR TRANSACTION;';
```

Direct Discovery-Zeichenvariablen

DirectFieldColumnDelimiter
Sie können für Datenbanken, die als Feldtrennzeichen ein anderes Zeichen als ein Komma erfordern, das in **Direct Query**-Befehlen als Feldtrennzeichen verwendete Zeichen festlegen. Das angegebene Zeichen muss im Befehl **SET** von einfachen Anführungszeichen umgeben sein.

```
SET DirectFieldColumnDelimiter= '|'
```

DirectStringQuoteChar
Sie können ein Zeichen angeben, das zum Anführen von Strings in einer generierten Abfrage zu verwenden ist. Standard ist ein einfaches Anführungszeichen. Das angegebene Zeichen muss im Befehl **SET** von einfachen Anführungszeichen umgeben sein.

```
SET DirectStringQuoteChar= '"';
```

DirectIdentifierQuoteStyle
Sie können angeben, dass in generierten Abfragen keine Anführung von Identifikatoren gemäß ANSI verwendet werden soll. Gleichzeitig ist die einzige verfügbare Anführung, die nicht den ANSI-Normen entspricht, GoogleBQ. Standard ist ANSI. Großschreibung, Kleinschreibung und gemischte Schreibung kann verwendet werden (ANSI, ansi, Ansi).

```
SET DirectIdentifierQuoteStyle="GoogleBQ";
```
Zum Beispiel wird eine Anführung gemäß ANSI im folgenden **SELECT**-Befehl verwendet:

```
SELECT [Quarter] FROM [qvTest].[sales] GROUP BY [Quarter]
```
Wenn **DirectIdentifierQuoteStyle** auf "GoogleBQ" gesetzt ist, würde der **SELECT**-Befehl folgende Anführung verwenden:

```
SELECT [Quarter] FROM [qvTest.sales] GROUP BY [Quarter]
```

DirectIdentifierQuoteChar
Sie können ein Zeichen angeben, um das Anführen von Identifikatoren in einer generierten Abfrage zu kontrollieren. Dieses kann entweder als ein Zeichen (wie ein doppeltes Anführungszeichen) oder zwei (wie ein Paar eckiger Klammern) festgelegt sein. Standard ist ein doppeltes Anführungszeichen.

```
SET DirectIdentifierQuoteChar='[]';
SET DirectIdentifierQuoteChar='``';
SET DirectIdentifierQuoteChar=' ';
SET DirectIdentifierQuoteChar='""';
```

DirectTableBoxListThreshold
Bei Verwendung von Direct Discovery-Felder in einer **Tabellen**visualisierung wird ein Schwellenwert festgelegt, der die Anzahl der angezeigten Reihen begrenzt. Standardschwellenwert ist 1000 Datensätze. Die Festlegung des Standardschwellenwerts kann durch Festsetzen der **DirectTableBoxListThreshold**-Variablen im Ladeskript geändert werden. Hier ein Beispiel:

```
SET DirectTableBoxListThreshold=5000;
```
Die Festlegung des Schwellenwerts gilt nur für **Tabellen**visualisierungen, die Direct Discovery-Felder enthalten. **Tabellen**visualisierungen, die nur im Speicher befindliche Felder enthalten, werden durch die **DirectTableBoxListThreshold**-Festsetzung nicht begrenzt.

In der **Tabellen**visualisierung werden solange keine Felder angezeigt, bis die Auswahl weniger Datensätze hat als die Schwellenwertgrenze.

Variable zur Interpretation von Direct Discovery-Zahlen

DirectMoneyDecimalSep
Das definierte Dezimaltrennzeichen ersetzt das Dezimalsymbol für Währung in dem SQL-Befehl, der für das Laden von Daten mithilfe von Direct Discovery generiert worden ist. Dieses Zeichen muss zum Zeichen passen, das in **DirectMoneyFormat** verwendet wurde.

Der Standardwert ist `'.'`

Beispiel:

```
Set DirectMoneyDecimalSep='.';
```

DirectMoneyFormat
Das definierte Symbol ersetzt das Währungsformat in dem SQL-Befehl, der für das Laden von Daten mithilfe von Direct Discovery generiert worden ist. Das Währungssymbol für das Tausendertrennzeichen sollte nicht eingeschlossen sein.

Der Standardwert ist `'#.0000'`

Beispiel:

```
Set DirectMoneyFormat='#.0000';
```

DirectTimeFormat
Das definierte Symbol ersetzt das Währungsformat in dem SQL-Befehl, der für das Laden von Daten mithilfe von Direct Discovery generiert worden ist.

Beispiel:

```
Set DirectTimeFormat='hh:mm:ss';
```

DirectDateFormat
Das definierte Symbol ersetzt das Währungsformat in dem SQL-Befehl, der für das Laden von Daten mithilfe von Direct Discovery generiert worden ist.

Beispiel:

```
Set DirectDateFormat='MM/DD/YYYY';
```

DirectTimeStampFormat
Das definierte Format ersetzt das Datum- und Uhrzeitformate in dem SQL-Befehl, der für das Laden von Daten mithilfe von Direct Discovery generiert worden ist.

Beispiel:

```
Set DirectTimestampFormat='M/D/YY hh:mm:ss[.fff]';
```

4.10 Fehlervariablen

Die Werte aller Fehlervariablen bleiben nach Ausführung des Skripts erhalten. Die erste Variable, ErrorMode, ist die Benutzereingabe, die letzten drei sind die Ausgabe von Qlik Sense mit Informationen zu den Fehlern im Skript.

Fehlervariablen – Übersicht

Die einzelnen Variablen werden nach der Übersicht genauer beschrieben. Sie können auch auf den Variablennamen in der Syntax klicken, um direkt auf die Details der betreffenden Variablen zuzugreifen.

In der Online-Hilfe von Qlik Sense finden Sie weitere Details zu der Variablen.

ErrorMode
Diese Fehlervariable legt fest, welche Aktion durch Qlik Sense ausgeführt werden soll, wenn ein Fehler während der Skriptausführung auftritt.

```
ErrorMode
```

ScriptError
Diese Fehlervariable liefert den Fehlercode des zuletzt ausgeführten Skriptbefehls.

```
ScriptError
```

ScriptErrorCount
Diese Fehlervariable liefert die Gesamtzahl der während der Skriptausführung gefundenen Fehler. Zu Beginn jeder Skriptausführung wird die Variable auf 0 zurückgesetzt.

```
ScriptErrorCount
```

ScriptErrorList
Diese Fehlervariable liefert eine Liste aller während der Skriptausführung aufgetretenen Fehler. Die Fehler sind durch Zeilenschaltung getrennt.

```
ScriptErrorList
```

ErrorMode

Diese Fehlervariable legt fest, welche Aktion durch Qlik Sense ausgeführt werden soll, wenn ein Fehler während der Skriptausführung auftritt.

Syntax:

```
ErrorMode
```

Argumente:

Argumente

Argument	Beschreibung
ErrorMode=1	Die Standardeinstellung. Die Ausführung des Skripts wird unterbrochen, der Fehler erscheint in einer Warnmeldung und der Anwender klickt auf OK oder Abbrechen (Non-Batch-Modus).
ErrorMode =0	Qlik Sense ignoriert den Fehler und die Skriptausführung wird beim nächsten Skriptbefehl fortgesetzt.
ErrorMode =2	Qlik Sense gibt bei einem Fehler unmittelbar die Meldung „Fehler bei der Ausführung des Skripts“ aus, ohne dass der Anwender zuvor eingreifen kann.

Beispiel:

```
set ErrorMode=0;
```

ScriptError

Diese Fehlervariable liefert den Fehlercode des zuletzt ausgeführten Skriptbefehls.

Syntax:

```
ScriptError
```

Nach jedem fehlerfrei ausgeführten Skriptbefehl wird die Variable auf 0 zurückgesetzt. Jedem Fehler wird ein interner Qlik Sense-Fehlercode zugewiesen, der aus einer Zahl und einem Text besteht. Die Fehlercodes lauten wie folgt:

Skriptfehlercodes

Fehlercode	Beschreibung
0	Kein Fehler. Der Text für den dualen Wert ist leer.
1	Allgemeiner Fehler.
2	Syntaxfehler.
3	Allgemeiner ODBC-Fehler.
4	Allgemeiner OLE DB-Fehler.
5	Allgemeiner benutzerdefinierter Datenbankfehler.
6	Allgemeiner XML-Fehler.
7	Allgemeiner HTML-Fehler.
8	Datei nicht gefunden.
9	Datenbank nicht gefunden.
10	Tabelle nicht gefunden.
11	Feld nicht gefunden.
12	Falsches Dateiformat.
16	Semantischer Fehler.

Beispiel:

```
set ErrorMode=0;

LOAD * from abc.qvf;
```

```
if ScriptError=8 then

exit script;

//no file;

end if
```

ScriptErrorCount

Diese Fehlervariable liefert die Gesamtzahl der während der Skriptausführung gefundenen Fehler. Zu Beginn jeder Skriptausführung wird die Variable auf 0 zurückgesetzt.

Syntax:

```
ScriptErrorCount
```

ScriptErrorList

Diese Fehlervariable liefert eine Liste aller während der Skriptausführung aufgetretenen Fehler. Die Fehler sind durch Zeilenschaltung getrennt.

Syntax:

```
ScriptErrorList
```

5 Formeln im Skript

Formeln können in den Befehlen **LOAD** und **SELECT** benutzt werden. Die hier beschriebene Syntax und die Funktionen beziehen sich auf den **LOAD**-Befehl, nicht auf den **SELECT**-Befehl. Letzterer wird vom ODBC-Treiber und nicht von Qlik Sense interpretiert. Die meisten ODBC-Treiber sind jedoch häufig in der Lage, einige der unten aufgeführten Funktionen zu interpretieren.

Formeln bestehen aus Funktionen, Feldern und Operatoren, die in einer Syntax kombiniert sind.

Alle Formeln in einem Qlik Sense-Skript geben eine Zahl und/oder eine Zeichenfolge zurück. Logische Funktionen und Operatoren liefern 0 für False und -1 für True. Konvertierungsmöglichkeiten zwischen Zahlen und Strings sind integriert. Logische Operatoren und Funktionen interpretieren 0 als False und alles andere als True.

Die allgemeine Syntax von Formeln lautet:

Allgemeine Syntax

Formel	**Felder**	**Operator**
expression ::= (constant	constant	|
expression ::= (constant	fieldref	|
expression ::= (constant	operator1 expression	|
expression ::= (constant	expression operator2 expression	|
expression ::= (constant	function	|
expression ::= (constant	(expression)	)

Dabei gilt:

- **constant** ist ein String (Text, Datum oder Uhrzeit) in einfachen geraden Anführungszeichen oder eine Zahl. Konstanten werden ohne Tausendertrennzeichen und mit einem Punkt als Dezimaltrennzeichen geschrieben.
- **fieldref** ist der Name eines Feldes der geladenen Tabelle.
- **operator1** ist ein einwertiger Operator. Er bezieht sich auf eine einzige Formel, die rechts vom Operator steht.
- **operator2** ist ein zweiwertiger Operator. Er bezieht sich auf zwei Formeln, eine links und eine rechts vom Operator.
- **function ::= functionname(parameters)**
- **parameters ::= expression { , expression }**

Zahl und Art der Parameter sind nicht willkürlich, Sie hängen von der verwendeten Funktion ab.

Formeln und Funktionen dagegen können beliebig angeordnet werden, solange das Ergebnis eindeutig ist. Qlik Sense gibt keine Fehlermeldungen aus.

6 Diagrammformeln

Eine Diagrammformel (Visualisierung) ist eine Kombination aus Funktionen, Feldern und mathematischen Operatoren (+ * / =) sowie anderen Kennzahlen. Formeln werden zum Verarbeiten von Daten in der App verwendet. Dabei wird ein Ergebnis ausgegeben, das in der Visualisierung veranschaulicht werden kann. Sie sind nicht auf die Verwendung in Kennzahlen beschränkt. Sie können dynamischere und leistungsfähigere Visualisierungen mit Formeln für Titel, Untertitel, Fußnoten und sogar Dimensionen erstellen.

Das bedeutet beispielsweise, dass für den Titel der Visualisierung anstatt statischem Text eine Formel verwendet werden kann, deren Ergebnis abhängig von den getroffenen Auswahlen unterschiedlich ausfallen kann.

Detaillierte Referenzen zur Skript- und Tabellenfunktionen finden Sie in der Datei Skriptsyntax und Diagrammfunktionen.

6.1 Definieren des Aggregierungsbereichs

Üblicherweise gibt es zwei Faktoren, die zusammen bestimmen, welche Datensätze zum Definieren des Aggregierungswerts in einer Formel verwendet werden. Beim Arbeiten in Visualisierungen sind diese Faktoren:

- Dimensionswert (der Aggregierung in einer Diagrammformel)
- Auswahl

Zusammen definieren diese Faktoren den Aggregierungsbereich. Es können Situationen auftreten, in denen Ihre Berechnung die Auswahl, Dimension oder beides außer Acht lassen soll. In Diagrammfunktionen erreichen Sie dies mit dem Qualifizierer TOTAL, der Aggregierung mit Auswahlformeln oder einer Kombination aus beidem.

Aggregierung: Methode und Beschreibung

Methode	Beschreibung
Qualifizierer TOTAL	Wenn Sie den Qualifizierer „Total“ in Ihrer Aggregierungsfunktion nutzen, wird der Dimensionswert nicht berücksichtigt. Die Aggregierung wird für alle wählbaren Feldwerte durchgeführt. Auf den Zusatz **TOTAL** kann eine Reihe von Feldnamen in spitzen Klammern folgen. Diese Feldnamen sollten eine Teilmenge der Dimensionen des Diagramms sein. der Funktion nur noch die explizit aufgeführten Dimensionen berücksichtigt, d. h. für jede Kombination von Werten dieser Felder wird ein Formelwert berechnet. Es können auch Felder aufgeführt werden, die nicht Dimension des Diagramms sind. Dies ist sinnvoll, wenn Gruppen als Dimension dienen. Führt man alle Variablen der Gruppe auf, kommt dies erst beim Wechsel des Drilldown zum Tragen.
Aggregierung mit Auswahlformeln	Wenn Sie die Aggregierung mit Auswahlformeln in Ihrer Aggregierung verwenden, wird die Auswahl aufgehoben. Die Aggregierung wird für alle Werte verteilt auf die Dimensionen durchgeführt.
TOTAL-Qualifizierer und Aggregierung mit Auswahlformeln	Wenn Sie den Qualifizierer **TOTAL** und die Aggregierung mit Auswahlformeln in Ihrer Aggregierung verwenden, wird die Auswahl aufgehoben und die Dimensionen werden nicht berücksichtigt.

Qualifizierer ALL	Wenn Sie den Qualifizierer **ALL** in Ihrer Aggregierung verwenden, werden die Auswahl und die Dimensionen nicht berücksichtigt. Dasselbe lässt sich mit der Anweisung für die Aggregierung mit Auswahlformeln {1} und dem Qualifizierer **TOTAL** erreichen:
	
	`=sum(All Sales)`
	
	`=sum({1} Total Sales)`

Beispiel: TOTAL-Qualifizierer

Im folgenden Beispiel wird gezeigt, wie Sie mit dem Qualifizierer TOTAL einen relativen Anteil berechnen können. Angenommen, Q2 wurde ausgewählt, dann wird mit TOTAL die Summe aller Werte unabhängig von den Dimensionen berechnet.

Beispiel: Total-Qualifizierer

Year	Quarter	Sum(Amount)	Sum(TOTAL Amount)	Sum(Amount)/Sum(TOTAL Amount)
		3000	3000	100%
2012	Q2	1700	3000	56,7%
2013	Q2	1300	3000	43,3%

*So zeigen Sie Zahlen als Prozentsatz an: Wählen Sie im Eigenschaftsfenster für die Kennzahl, die Sie als Prozentwert anzeigen möchten, unter **Zahlenformat** die Option **Zahl** und unter **Formatierung** die Option **Standard** und eines der %-Formate aus.*

Beispiel: Aggregierung mit Auswahlformeln

In dem folgenden Beispiel wird gezeigt, wie Sie mit der Aggregierung mit Auswahlformeln einen Vergleich zwischen Datensätzen anstellen können, bevor Auswahlen getroffen wurden. Angenommen, Q2 wurde ausgewählt. Dann berechnet die Aggregierung mit Auswahlformeln mit der Auswahlfunktion {1} die Summe aller Werte unabhängig von der Auswahl, jedoch verteilt auf die Dimensionen.

Beispiel: Aggregierung mit Auswahlformeln

Year	Quarter	Sum(Amount)	Sum({1} Amount)	Sum(Amount)/Sum({1} Amount)
		3000	10800	27,8%
2012	Q1	0	1100	0%
2012	Q3	0	1400	0%
2012	Q4	0	1800	0%
2012	Q2	1700	1700	100%
2013	Q1	0	1000	0%
2013	Q3	0	1100	0%
2013	Q4	0	1400	0%
2013	Q2	1300	1300	100%

Beispiel: TOTAL-Zusatz und Aggregierung mit Auswahlformeln

Im folgenden Beispiel wird gezeigt, wie Sie mit der Aggregierung mit Auswahlformeln und dem Qualifizierer TOTAL einen Vergleich zwischen Datensätzen anstellen können – bevor eine Auswahl getroffen wurde und für alle Dimensionen. Angenommen, Q2 wurde mit Aggregierung mit Auswahlformeln mit der Definition für Auswahlformeln {1} und dem Qualifizierer TOTAL ausgewählt und berechnet die Summe aller Werte, indem alle Auswahlen und alle Dimensionen verworfen werden.

Beispiel: TOTAL-Zusatz und Aggregierung mit Auswahlformeln

Year	Quarter	Sum (Amount)	Sum({1} TOTAL Amount)	Sum(Amount)/Sum({1} TOTAL Amount)
		3000	10800	27,8%
2012	Q2	1700	10800	15,7%
2013	Q2	1300	10800	12%

In Beispielen verwendete Daten:

```
AggregationScope:
LOAD * inline [
Year Quarter Amount
2012 Q1 1100
2012 Q2 1700
2012 Q3 1400
2012 Q4 1800
2013 Q1 1000
2013 Q2 1300
2013 Q3 1100
2013 Q4 1400] (delimiter is ' ');
```

6.2 Aggregierung mit Auswahlformeln

Wenn Sie eine Auswahl in einer App treffen, definieren Sie eine Teilmenge der Datensätze in den Daten. Aggregierungsfunktionen wie `Sum()`, `Max()`, `Min()`, `Avg()` und `Count()` werden anhand dieser Teilmenge berechnet.

Ihre Auswahl definiert also den Umfang der Aggregierung, den Satz der Datensätze, mit denen Berechnungen durchgeführt werden.

Aggregierung mit Auswahlformeln bietet eine Möglichkeit, einen Umfang zu definieren, der sich vom durch die aktuellen Auswahlen definierten Satz an Datensätzen unterscheidet. Dieser neue Umfang kann auch als alternative Auswahl betrachtet werden.

Das kann nützlich sein, wenn Sie die aktuelle Auswahl mit einem bestimmten Wert vergleichen möchten, beispielsweise dem Wert des letzten Jahres oder dem globalen Marktanteil.

Auswahlformeln

Auswahlformeln können inner- und außerhalb von Aggregierungsfunktionen verwendet werden und sind in geschweifte Klammern eingeschlossen.

Beispiel: Innere Auswahlformel

```
Sum( {$<Year={2021}>} Sales )
```

Beispiel: Äußere Auswahlformel

```
{<Year={2021}>} Sum(Sales) / Count(distinct Customer)
```

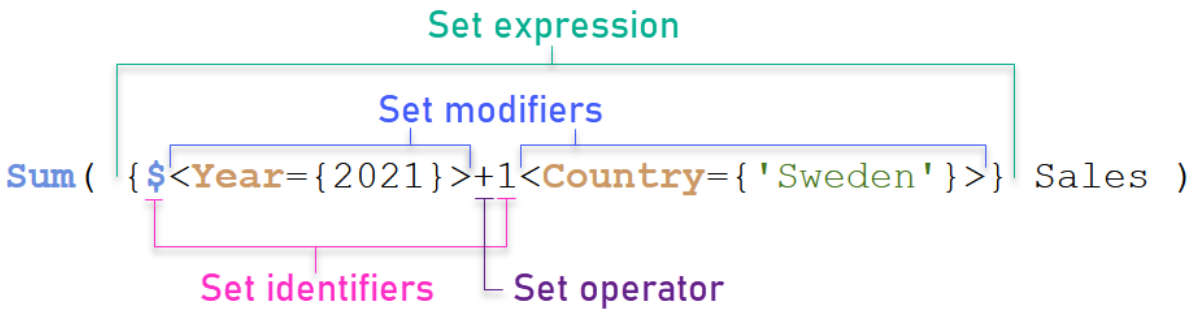
Eine Auswahlformel besteht aus einer Kombination der folgenden Elemente:

- **Identifikatoren**. Ein Identifikator für Auswahlformeln stellt eine Auswahl dar, die anderswo definiert wurde. Er stellt auch eine bestimmte Gruppe Datensätze in den Daten dar. Es kann sich um die aktuelle Auswahl, eine Auswahl durch ein Lesezeichen oder eine Auswahl durch einen alternativen Zustand handeln. Eine einfache Auswahlformel besteht aus einem einzelnen Identifikator, wie dem Dollarzeichen, {$}, was alle Datensätze in der aktuellen Auswahl bedeutet.
  Beispiele: `$`, `1`, `BookMark1`, `State2`

- **Operatoren**. Ein Operator für Auswahlformeln kann verwendet werden, um Verbindungen, Unterschiede oder Überschneidungen zwischen verschiedenen Identifikatoren für Auswahlformeln zu erstellen. Auf diese Weise können Sie eine Teilmenge oder eine Obermenge der durch die Identifikatoren für Auswahlformeln definierten Auswahlen erstellen.
  Beispiele: +, -, *, /
- **Modifikatoren**. Ein Modifikator für Auswahlformeln kann zu einem Identifikator für Auswahlformeln hinzugefügt werden, um dessen Auswahl zu ändern. Ein Modifikator kann auch für sich selbst verwendet werden und ändert in diesem Fall den Standardidentifikator. Ein Modifikator muss in spitze Klammern eingeschlossen werden (<…>).
  Beispiele: `<Year={2020}>`, `<Supplier={ACME}>`

Die Elemente werden so kombiniert, dass sie Auswahlformeln bilden.

Elemente in einer Auswahlformel



Die obige Auswahlformel wird beispielsweise anhand der Aggregierung `Sum(Sales)` erstellt.

Der erste Operand gibt Umsätze für das Jahr `2021` für die aktuelle Auswahl zurück, was durch den Identifikator für Auswahlformeln `$` und den Modifikator angegeben wird, der die Auswahl des Jahres `2021` enthält. Der zweite Operand gibt `Sales` für `Sweden` zurück und ignoriert die aktuelle Auswahl, was durch den Identifikator für Auswahlformeln `1` angegeben wird.

Schließlich gibt die Formel einen Satz bestehend aus den Datensätzen zurück, die zu einem beliebigen der beiden Operanden gehören, wie durch den Operanden + angegeben.

Beispiele

Beispiele, in denen die obigen Elemente für Auswahlformeln kombiniert werden, finden Sie in den folgenden Themen:

Natürliche Sätze

In der Regel steht eine Auswahlformel sowohl für einen Satz Datensätze im Datenmodell als auch für eine Auswahl, welche diese Teilmenge der Daten definiert. In diesem Fall wird die Gruppe als natürlicher Satz bezeichnet.

Identifikatoren für Auswahlformeln, mit oder ohne Modifikatoren für Auswahlformeln, stellen immer natürliche Sätze dar.

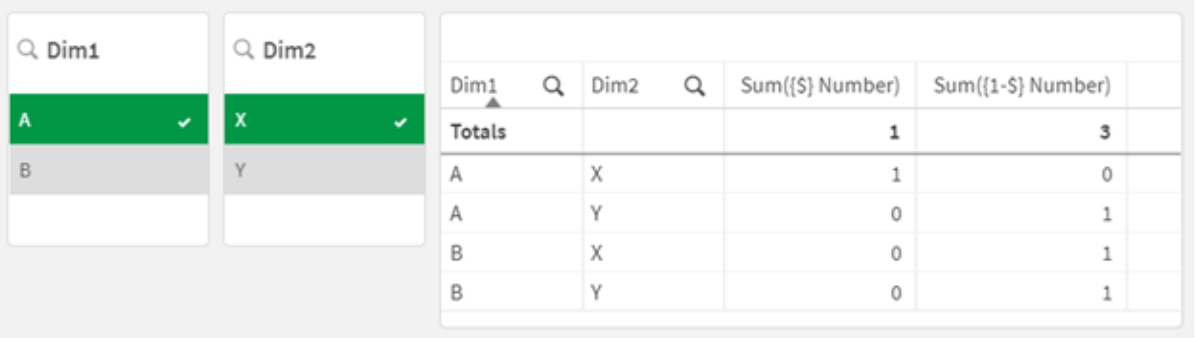
Eine Auswahlformel, die Operatoren für Auswahlformeln verwendet, kann zwar ebenfalls eine Teilmenge der Datensätze darstellen, kann aber in der Regel nicht anhand einer Auswahl von Feldwerten beschrieben werden. Eine solche Formel ist ein nicht natürlicher Satz.

Beispielsweise kann der von `{1-$}` angegebene Satz nicht immer durch eine Auswahl definiert werden. Daher ist er kein natürlicher Satz. Dies kann gezeigt werden, indem die folgenden Daten geladen und zu einer Tabelle hinzugefügt werden und dann Auswahlen mithilfe von Filterfenstern getroffen werden.

```
Load * Inline
[Dim1, Dim2, Number
A, X, 1
A, Y, 1
B, X, 1
B, Y, 1];
```

Indem Sie Auswahlen für `Dim1` und `Dim2` treffen, erhalten Sie die in der folgenden Tabelle gezeigte Ansicht.

Tabellen mit natürlichen und nicht natürlichen Sätzen



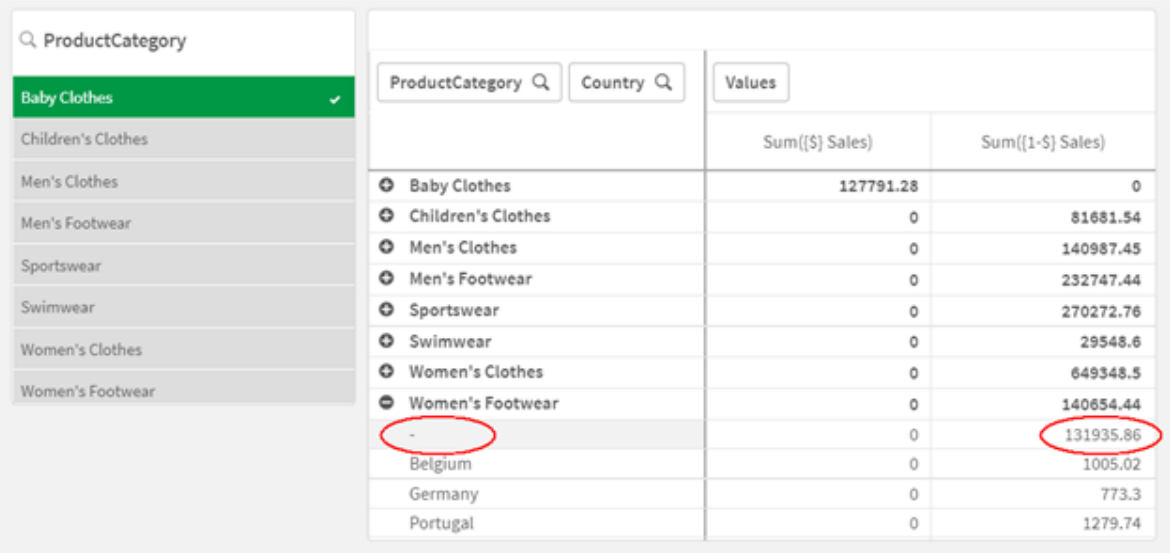
Die Auswahlformel in der ersten Kennzahl verwendet einen natürlichen Satz: Sie entspricht der Auswahl, die in `{$}` getroffen wird.

Die zweite Kennzahl ist anders. Sie verwendet `{1-$}`. Es ist nicht möglich, eine Auswahl zu treffen, die diesem Satz entspricht, also handelt es sich um einen nicht natürlichen Satz.

Diese Unterscheidung hat eine Reihe von Folgen:

- Modifikatoren für Auswahlformeln können nur auf Identifikatoren für Auswahlformeln angewendet werden. Sie können nicht auf eine beliebige Auswahlformel angewendet werden. Beispielsweise ist es nicht möglich, eine Auswahlformel wie die folgende zu verwenden:
  `{ (BM01 * BM02) <Field={x,y}> }`
  Hier geben die normalen (runden) Klammern an, dass die Überscheidung zwischen `BM01` und `BM02` ausgewertet werden muss, bevor der Modifikator für Auswahlformeln angewendet wird. Der Grund ist, dass kein Elementsatz vorhanden ist, der geändert werden kann.
- Sie können keine nicht natürlichen Sätze innerhalb von `P()`- und `E()`-Elementfunktionen verwenden. Diese Funktionen geben einen Elementsatz zurück, aber es ist nicht möglich, den Elementsatz aus einem nicht natürlichen Satz herzuleiten.

- Eine Kennzahl, die einen nicht natürlichen Satz verwendet, kann nicht immer dem richtigen Dimensionswert zugeordnet werden, wenn das Datenmodell viele Tabellen enthält. Beispielsweise werden im folgenden Diagramm einige ausgeschlossene Umsatzzahlen dem richtigen Country zugeordnet, während andere NULL als Country aufweisen.
  Diagramm mit nicht natürlichem Satz



  Ob die Zuweisung korrekt erfolgt oder nicht, hängt vom Datenmodell ab. In diesem Fall kann die Zahl nicht zugewiesen werden, wenn sie zu einem Land gehört, das durch die Auswahl ausgeschlossen wurde.

Identifikator	Beschreibung
1	Stellt den vollständigen Satz aller Datensätze in der Anwendung dar, unabhängig von den vorgenommenen Auswahlen.
$	Stellt die Datensätze der aktuellen Auswahl dar. Die Auswahlformel **{$}** bezeichnet die aktuelle Auswahl und kann somit auch weggelassen werden.
$1	Stellt die vorherige Auswahl dar. $2 stellt die vorletzte Auswahl dar und so weiter.
$_1	Stellt die nächste (vorwärts) Auswahl dar. $_2 stellt die übernächste Auswahl dar und so weiter.
BM01	Sie können eine Lesezeichen-ID oder einen Lesezeichennamen verwenden.
MyAltState	Sie können die in einem alternativen Status getroffenen Auswahlen nach dem Statusnamen referenzieren.

Beispiel	Ergebnis
sum ({1} Sales)	Liefert den Umsatz für alle Werte der App, unabhängig von den aktuellen Auswahlen, aber nicht von der Dimension.

Beispiel	Ergebnis
sum ({$} Sales)	Liefert den Umsatz für die aktuelle Auswahl, äquivalent zu sum(Sales).
sum ({$1} Sales)	Liefert den Umsatz für die vorherige Auswahl.
sum ({BM01} Sales)	Liefert den Umsatz für das Lesezeichen mit dem Namen *BM01*.

Beispiel	Ergebnis
sum({$<OrderDate = DeliveryDate>} Sales)	Liefert den Umsatz für die aktuelle Auswahl OrderDate = DeliveryDate.
sum({1<Region = {US}>} Sales)	Liefert den Umsatz für die Region USA, ohne die aktuelle Auswahl zu berücksichtigen.
sum({$<Region = >} Sales)	Liefert den Umsatz für die Auswahl, aber ohne die Auswahl im Feld *Region*.
sum({<Region = >} Sales)	Entspricht dem Beispiel darüber. Wird kein Modifikator verwendet, wird $ angenommen.
sum({$<Year={2000}, Region={“U*”}>} Sales)	Liefert den Umsatz für die aktuelle Auswahl, jedoch mit neuen Auswahlen in *Year* und *Region*.

Identifikatoren für Auswahlformeln

Ein Identifikator für Auswahlformeln stellt einen Satz Datensätze in den Daten dar, entweder in allen Daten oder in einer Teilmenge der Daten. Es handelt sich um einen Satz Datensätze, der durch eine Auswahl definiert wird. Es kann sich um die aktuelle Auswahl, alle Daten (keine Auswahl), eine Auswahl durch ein Lesezeichen oder eine Auswahl durch einen alternativen Zustand handeln.

Im Beispiel `Sum( {$<Year = {2009}>} Sales )` ist der Identifikator das Dollarzeichen: `$`. Dies stellt die aktuellen Auswahl dar. Es stellt auch alle möglichen Datensätze dar. Dieser Satz kann dann durch den Modifikatorteil der Auswahlformel weiter geändert werden: Die Auswahl `2009` in `Year` wird hinzugefügt.

In einer komplexeren Auswahlformel können zwei Identifikatoren zusammen mit einem Operator verwendet werden, um eine Verbindung, einen Unterschied oder eine Überschneidung der beiden Datensätze zu bilden.

Die folgende Tabelle zeigt einige häufige Identifikatoren.

Beispiele mit häufigen Identifikatoren

Identifikator	Beschreibung
1	Stellt den vollständigen Satz aller Datensätze in der Anwendung dar, unabhängig von den vorgenommenen Auswahlen.

Identifikator	Beschreibung
`$`	Stellt die Datensätze der aktuellen Auswahl im Standardzustand dar. Wenn die Auswahlformel `{$}` angegeben wird, entspricht dies also der Angabe keiner Auswahlformel.
`$1`	Stellt die vorherige Auswahl im Standardzustand dar. `$2` stellt die vorletzte Auswahl dar, usw.
`$_1`	Stellt die nächste Auswahl (in Vorwärtsrichtung) dar. `$_2` stellt die übernächste Auswahl dar, usw.
`BM01`	Sie können eine Lesezeichen-ID oder einen Lesezeichennamen verwenden.
`AltState`	Sie können einen alternativen Zustand nach dem Zustandsnamen referenzieren.
`AltState::BM01`	Ein Lesezeichen enthält die Auswahl aller Zustände, und Sie können ein spezifisches Lesezeichen referenzieren, indem Sie den Lesezeichennamen qualifizieren.

Die folgende Tabelle zeigt Beispiele mit verschiedenen Identifikatoren.

Beispiele mit verschiedenen Identifikatoren

Beispiel	Ergebnis
`Sum ({1} Sales)`	Liefert den Umsatz für alle Werte der App, unabhängig von den Auswahlen, aber nicht von der Dimension.
`Sum ({$} Sales)`	Liefert den Umsatz für die aktuelle Auswahl, äquivalent zu `Sum (Sales)`.
`Sum ({$1} Sales)`	Liefert den Umsatz für die vorherige Auswahl.
`Sum ({BM01} Sales)`	Liefert den Umsatz für das Lesezeichen mit dem Namen `BM01`.

Operatoren für Auswahlformeln

Operatoren für Auswahlformeln werden zum Einschließen, Ausschließen und Überschneiden von Datensätzen verwendet. Sie benutzen Auswahlformeln als Operanden und liefern wiederum eine Auswahlformel.

Sie können Operatoren für Auswahlformeln in verschiedenen Situationen einsetzen:

- Zum Durchführen eines Vorgangs für Identifikatoren für Auswahlformeln, der Sätze von Datensätzen in Daten darstellt.
- Zum Durchführen eines Vorgangs für die Elementsätze, für die Feldwerte oder innerhalb eines Modifikators für Auswahlformeln.

Die folgende Tabelle zeigt Operatoren, die in Auswahlformeln verwendet werden können.

Operatoren

Operator	Beschreibung
+	Vereinigung. Diese zweiwertige Operation liefert einen Satz aus Datensätzen oder Elementen, die zu mindestens einem der beiden Operanden gehören.
-	Ausschluss. Diese zweiwertige Operation liefert einen Satz aus Datensätzen oder Elemente, die zum ersten, nicht aber zum zweiten Operanden gehören. Als einwertiger Operator liefert er die Umkehrung der momentanen Auswahl.
*	Schnittmenge. Diese zweiwertige Operation liefert einen Satz aus Datensätzen oder Elemente, die zu beiden Operanden gehören.
/	Symmetrische Differenz (XOR). Diese zweiwertige Operation liefert einen Satz aus Datensätzen oder Elementen, die zu einem der Operanden, aber nicht zu beiden gehören.

Die folgende Tabelle zeigt Beispiele mit Operatoren.

Beispiele mit Operatoren

Beispiel	Ergebnis
`Sum ({1-$} Sales)`	Liefert den Umsatz für alle von der aktuellen Auswahl ausgeschlossenen Werte
`Sum ({$*BM01} Sales)`	Liefert den Umsatz für die Schnittmenge zwischen der Auswahl und Lesezeichen #160;BM01.
`Sum ({-($+BM01)} Sales)`	Liefert den Umsatz für die Werte, die durch die Auswahl und Lesezeichen BM01 ausgeschlossen sind.
`Sum ({$<Year= {2009}>+1<Country= {'Sweden'}>} Sales)`	Liefert den Umsatz für das Jahr 2009, der mit den aktuellen Auswahlen verknüpft ist, und fügt alle Daten hinzu, die mit dem Land Sweden über alle Jahre verknüpft sind.
`Sum ({$<Country= {"S*"}+{"*land"}>} Sales)`	Liefert den Umsatz für die Länder, die mit S beginnen oder mit land enden.

Modifikatoren für Auswahlformeln

Auswahlformeln werden verwendet, um den Umfang einer Berechnung zu definieren. Der zentrale Teil der Auswahlformel ist der Modifikator für Auswahlformeln, der eine Auswahl angibt. Dies wird verwendet, um die Benutzerauswahl bzw. die Auswahl im Identifikator für Auswahlformeln zu modifizieren, und das Ergebnis definiert einen neuen Umfang für die Berechnung.

Der Modifikator für Auswahlformeln besteht aus mindestens einem Feldnamen, jeweils gefolgt von einer Auswahl von Feldwerten. Der Modifikator ist in spitze Klammern eingeschlossen: < >

Hier ein Beispiel:

- `Sum ( {$<Year = {2015}>} Sales )`
- `Count ( {1<Country = {Germany}>} distinct OrderID )`

- `Sum ( {$<Year = {2015}, Country = {Germany}>} Sales )`

Elementsätze

Ein Elementsatz kann wie folgt definiert werden:

- Mit einer Werteliste
- Mit einer Suche
- Mit einem Verweis auf ein anderes Feld
- Mit einer Auswahlfunktion

Wenn die Definition des Elementsatzes ausgelassen wird, löscht der Modifikator für Auswahlformeln jede Auswahl in diesem Feld. Hier ein Beispiel:

```
Sum( {$<Year = >} Sales )
```

Beispiele: Diagrammformeln für Modifikatoren für Auswahlformeln, basierend auf Elementsätzen

Beispiele – Diagrammformeln

Ladeskript

Laden Sie die folgenden Daten als Inline-Ladevorgang in den Dateneditor, um die folgenden Diagrammformelbeispiele zu erstellen.

```
MyTable:
Load * Inline [
Country, Year, Sales
Argentina, 2014, 66295.03
Argentina, 2015, 140037.89
Austria, 2014, 54166.09
Austria, 2015, 182739.87
Belgium, 2014, 182766.87
Belgium, 2015, 178042.33
Brazil, 2014, 174492.67
Brazil, 2015, 2104.22
Canada, 2014, 101801.33
Canada, 2015, 40288.25
Denmark, 2014, 45273.25
Denmark, 2015, 106938.41
Finland, 2014, 107565.55
Finland, 2015, 30583.44
France, 2014, 115644.26
France, 2015, 30696.98
Germany, 2014, 8775.18
Germany, 2015, 77185.68
];
```

Diagrammformeln

Erstellen Sie eine Tabelle in einem Qlik Sense Arbeitsblatt mit den folgenden Diagrammformeln.

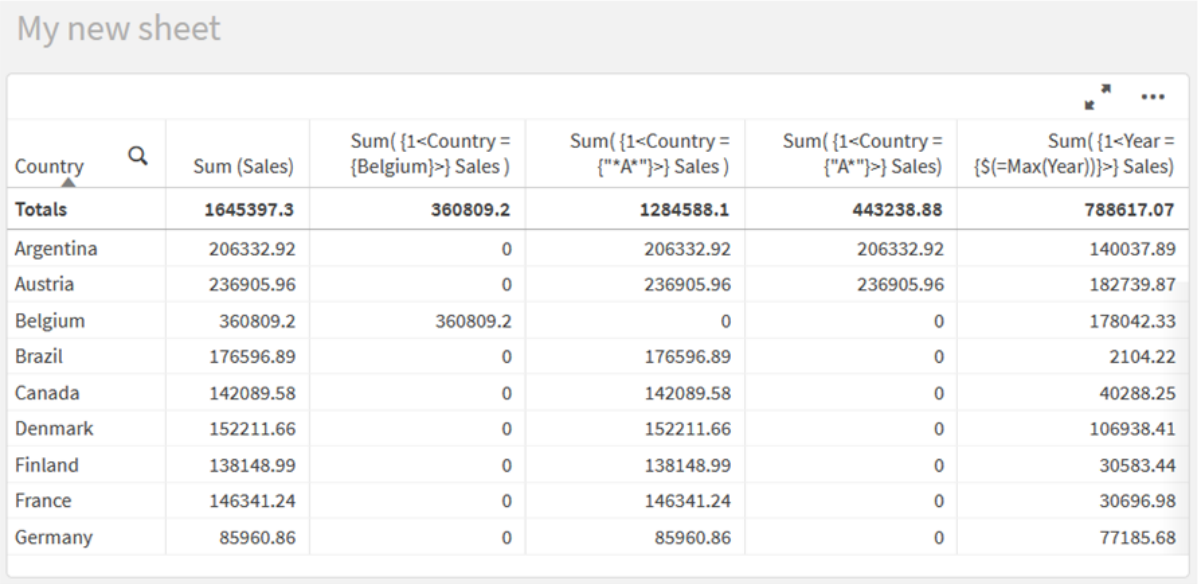
Tabelle – auf Elementsätzen basierende Modifikatoren für Auswahlformeln

Land	**Sum(Sales)**	**Sum ({1<Country= {Belgium}>} Sales)**	**Sum ({1<Country= {"*A*"}>} Sales)**	**Sum ({1<Country= {"A*"}>} Sales)**	**Sum({1<Year= {$(=Max (Year))}>} Sales)**
Summen	1645397.3	360809.2	1284588.1	443238.88	788617.07
Argentinien	206332.92	0	206332.92	206332.92	140037.89
Österreich	236905.96	0	236905.96	236905.96	182739.87
Belgien	360809.2	360809.2	0	0	178042.33
Brasilien	176596.89	0	176596.89	0	2104.22
Kanada	142089.58	0	142089.58	0	40288.25
Dänemark	152211.66	0	152211.66	0	106938.41
Finnland	138148.99	0	138148.99	0	30583.44
Frankreich	146341.24	0	146341.24	0	30696.98
Deutschland	85960.86	0	85960.86	0	77185.68

Erläuterung

- Dimension:
  - `Country`
- Kennzahl:
  - `Sum(Sales)`
    `Sales` ohne Auswahlformel summieren.
  - `Sum({1<Country={Belgium}>}Sales)`
    `Belgium` auswählen und die zugehörigen `Sales` summieren.
  - `Sum({1<Country={"*A*"}>}Sales)`
    Alle Länder auswählen, die ein `A` enthalten, und die zugehörigen `Sales` summieren.
  - `Sum({1<Country={"A*"}>}Sales)`
    Alle Länder auswählen, die mit einem `A` beginnen, und die zugehörigen `Sales` summieren.
  - `Sum({1<Year={$(=Max(Year))}>}Sales)`
    Das `Max(Year)` berechnen, in diesem Fall `2015`, und dann die entsprechenden `Sales` summieren.

Auf Elementsätzen basierende Modifikatoren für Auswahlformeln


My new sheet

Country	Sum (Sales)	Sum({1<Country = {Belgium}>} Sales)	Sum({1<Country = {"*A*"}>} Sales)	Sum({1<Country = {"A*"}>} Sales)	Sum({1<Year = {$(=Max(Year))}>} Sales)
Totals	**1645397.3**	**360809.2**	**1284588.1**	**443238.88**	**788617.07**
Argentina	206332.92	0	206332.92	206332.92	140037.89
Austria	236905.96	0	236905.96	236905.96	182739.87
Belgium	360809.2	360809.2	0	0	178042.33
Brazil	176596.89	0	176596.89	0	2104.22
Canada	142089.58	0	142089.58	0	40288.25
Denmark	152211.66	0	152211.66	0	106938.41
Finland	138148.99	0	138148.99	0	30583.44
France	146341.24	0	146341.24	0	30696.98
Germany	85960.86	0	85960.86	0	77185.68

Aufgelistete Werte

Das gängigste Beispiel eines Elementsatzes ist ein Satz, der auf einer Liste von in geschweiften Klammern stehenden Feldwerten basiert. Hier ein Beispiel:

- `{$<Country = {Canada, Germany, Singapore}>}`
- `{$<Year = {2015, 2016}>}`

Die inneren geschweiften Klammern definieren den Elementsatz. Die einzelnen Werte werden durch Kommas getrennt.

Anführungszeichen und Unterscheidung nach Groß- und Kleinschreibung

Wenn die Werte Leer- oder Sonderzeichen enthalten, müssen die Werte in Anführungszeichen gesetzt werden. Einfache Anführungszeichen bezeichnen eine exakte Übereinstimmung mit einem einzelnen Feldwert mit Beachtung der Groß- und Kleinschreibung. Doppelte Anführungszeichen bezeichnen eine Übereinstimmung mit einem oder mehreren Feldwerten ohne Beachtung der Groß- und Kleinschreibung. Hier ein Beispiel:

- `<Country = {'New Zealand'}>`
  Entspricht nur `New Zealand`.
- `<Country = {"New Zealand"}>`
  Entspricht `New Zealand`, `NEW ZEALAND` und `new zealand`.

Datumsangaben müssen in Anführungszeichen eingeschlossen werden und das Datumsformat des betreffenden Felds verwenden. Hier ein Beispiel:

- `<ISO_Date = {'2021-12-31'}>`
- `<US_Date = {'12/31/2021'}>`
- `<UK_Date = {'31/12/2021'}>`

Doppelte Anführungszeichen können durch eckige Klammern oder Gravis-Zeichen ersetzt werden.

Suchvorgänge

Elementsätze können auch durch Suchen erstellt werden. Hier ein Beispiel:

- `<Country = {"C*"}>`
- `<Ingredient = {"*garlic*"}>`
- `<Year = {">2015"}>`
- `<Date = {">12/31/2015"}>`

In Textsuchen können Platzhalter verwendet werden: Ein Asterisk (*) steht für eine beliebige Anzahl von Zeichen, ein Fragezeichen (?) für ein einziges Zeichen. Zum Definieren numerischer Suchen können relationale Operationen verwendet werden.

Verwenden Sie für Suchen immer doppelte Anführungszeichen. Bei Suchvorgängen wird die Groß- und Kleinschreibung nicht berücksichtigt.

Dollarzeichenerweiterungen

Dollarzeichenerweiterungen sind erforderlich, wenn Sie eine Berechnung innerhalb Ihres Elementsatzes verwenden möchten. Beispiel: Wenn Sie nur das letzte mögliche Jahr anzeigen möchten, verwenden Sie:

```
<Year = {$(=Max(Year))}>
```

Ausgewählte Werte in anderen Feldern

Modifikatoren können auf den ausgewählten Werten eines anderen Felds basieren. Hier ein Beispiel:

```
<OrderDate = DeliveryDate>
```

Durch diesen Modifikator werden die ausgewählten Werte des Felds '`DeliveryDate`' als Auswahl im Feld '`OrderDate`' verwendet. Wenn dies viele distinkte Werte sind – mehrere Hundert –, erfordert diese Operation viel Rechenleistung und sollte vermieden werden.

Elementsatzfunktionen

Der Elementsatz kann auf den Auswahlfunktionen `P()` (mögliche Werte) und `E()` (ausgeschlossene Werte) basieren.

Beispiel: Wenn Sie Länder auswählen möchten, in denen das Produkt `Cap` verkauft wurde, können Sie Folgendes verwenden:

```
<Country = P({1<Product={Cap}>} Country)>
```

Wenn Sie die Länder auswählen möchten, in denen das Produkt `Cap` nicht verkauft wurde, können Sie Folgendes verwenden:

```
<Country = E({1<Product={Cap}>} Country)>
```

Modifikatoren für Auswahlformeln mit Suchen

Mit Modifikatoren für Auswahlformeln können Sie Elementsätze über Suchen erstellen.

Hier ein Beispiel:

- `<Country = {"C*"}>`
- `<Year = {">2015"}>`
- `<Ingredient = {"*garlic*"}>`

Suchen müssen immer in doppelte Anführungszeichen, eckige Klammern oder Graviszeichen eingeschlossen werden. Sie können eine Liste mit einer Kombination aus Literalzeichenfolgen (einfache Anführungszeichen) und Suchen (doppelte Anführungszeichen) verwenden. Hier ein Beispiel:

```
<Product = {'Nut', "*Bolt", Washer}>
```

Textsuchen

In Textsuchen können Platzhalter und andere Symbole verwendet werden:

- Ein Asterisk (*) steht für eine beliebige Anzahl Zeichen.
- Ein Fragezeichen (?) steht für ein einziges Zeichen.
- Ein Zirkumflex-Zeichen (^) markiert den Anfang eines Worts.

Hier ein Beispiel:

- `<Country = {"C*", "*land"}>`
  Alle Länder abgleichen, die mit einem `C` beginnen oder mit einem `land` enden.
- `<Country = {"*^z*"}>`
  Damit werden alle Länder abgeglichen, die ein Wort enthalten, das mit einem `z` beginnt, beispielsweise `New Zealand`.

Numerische Suchen

Sie können numerische Suchen anhand der folgenden relationalen Operatoren durchführen: >, >=, <, <=

Eine numerische Suche beginnt immer mit einem dieser Operatoren. Hier ein Beispiel:

- `<Year = {">2015"}>`
  2016 und nachfolgende Jahre abgleichen.
- `<Date = {">=1/1/2015<1/1/2016"}>`
  Jedes Datum während 2015 abgleichen. Beachten Sie die Syntax zum Beschreiben eines Zeitraums zwischen zwei Datumsangaben. Das Datumsformat muss dem Datumsformat des betreffenden Felds entsprechen.

Formelsuchen

Sie können Formelsuchen verwenden, um erweiterte Suchen durchzuführen. Eine Aggregierung wird dann für jeden Feldwert im Suchfeld ausgewertet. Alle Werte, bei denen die Suchformel „wahr“ zurückgibt, werden ausgewählt.

Eine Formelsuche beginnt immer mit einem Gleichheitszeichen: =

Hier ein Beispiel:

```
<Customer = {"=Sum(Sales)>1000"}>
```

Damit werden alle Kunden mit einem Umsatzwert von mehr als 1000 zurückgegeben. `Sum(Sales)` wird für die aktuelle Auswahl berechnet. Wenn Sie also eine Auswahl in einem anderen Feld haben, z. B. im Feld `Product`, erhalten Sie nur die Kunden, welche die Umsatzbedingung für die ausgewählten Produkte erfüllt haben.

Wenn die Bedingung unabhängig von der Auswahl sein soll, müssen Sie eine Aggregierung mit Auswahlformeln innerhalb der Suchzeichenfolge verwenden. Hier ein Beispiel:

```
<Customer = {"=Sum({1} Sales)>1000"}>
```

Die Formeln nach dem Gleichheitszeichen werden als boolescher Wert interpretiert. Wenn dann das Ergebnis ein anderer Wert ist, wird jede andere Zahl als Null als wahr interpretiert, während Null und Zeichenfolgen als falsch interpretiert werden.

Quotes

Verwenden Sie Anführungszeichen, wenn die Suchzeichenfolgen Leer- oder Sonderzeichen enthalten. Einfache Anführungszeichen geben eine exakte Übereinstimmung mit einem einzelnen Feldwert mit Beachtung der Groß- und Kleinschreibung an. Doppelte Anführungszeichen geben eine Suche ohne Beachtung der Groß- und Kleinschreibung an, die mit mehreren Feldwerten übereinstimmen kann.

Hier ein Beispiel:

- `<Country = {'New Zealand'}>`
  Entspricht nur `New Zealand`.
- `<Country = {"New Zealand"}>`
  Entspricht `New Zealand`, `NEW ZEALAND` und `new zealand`.

Doppelte Anführungszeichen können durch eckige Klammern oder Gravis-Zeichen ersetzt werden.

> *In früheren Versionen von Qlik Sense wurde nicht zwischen einfachen und doppelten Anführungszeichen unterschieden. Alle in Anführungszeichen gesetzten Strings wurden als Suchvorgänge behandelt. Aus Gründen der Abwärtskompatibilität funktionieren mit älteren Versionen von Qlik Sense erstellte Apps weiterhin wie in früheren Versionen. In Apps, die mit Qlik Sense November 2017 oder höher erstellt wurden, wird der Unterschied zwischen den beiden Typen von Anführungszeichen berücksichtigt.*

Beispiele: Diagrammformeln für Modifikatoren für Auswahlformen mit Suchen

Beispiele – Diagrammformeln

Ladeskript

Laden Sie die folgenden Daten als Inline-Ladevorgang in den Dateneditor, um die folgenden Diagrammformelbeispiele zu erstellen.

```
MyTable:
Load
Year(Date) as Year,
Date#(Date,'YYYY-MM-DD') as ISO_Date,
```

```
Date(Date#(Date,'YYYY-MM-DD'),'M/D/YYYY') as US_Date,
Country, Product, Amount
Inline
[Date, Country, Product, Amount
2018-02-20, Canada, Washer, 6
2018-07-08, Germany, Anchor bolt, 10
2018-07-14, Germany, Anchor bolt, 3
2018-08-31, France, Nut, 2
2018-09-02, Czech Republic, Bolt, 1
2019-02-11, Czech Republic, Bolt, 3
2019-07-31, Czech Republic, Washer, 6
2020-03-13, France, Anchor bolt, 1
2020-07-12, Canada, Anchor bolt, 8
2020-09-16, France, Washer, 1];
```

Beispiel 1: Diagrammformeln mit Textsuchen

Erstellen Sie eine Tabelle in einem Qlik Sense Arbeitsblatt mit den folgenden Diagrammformeln.

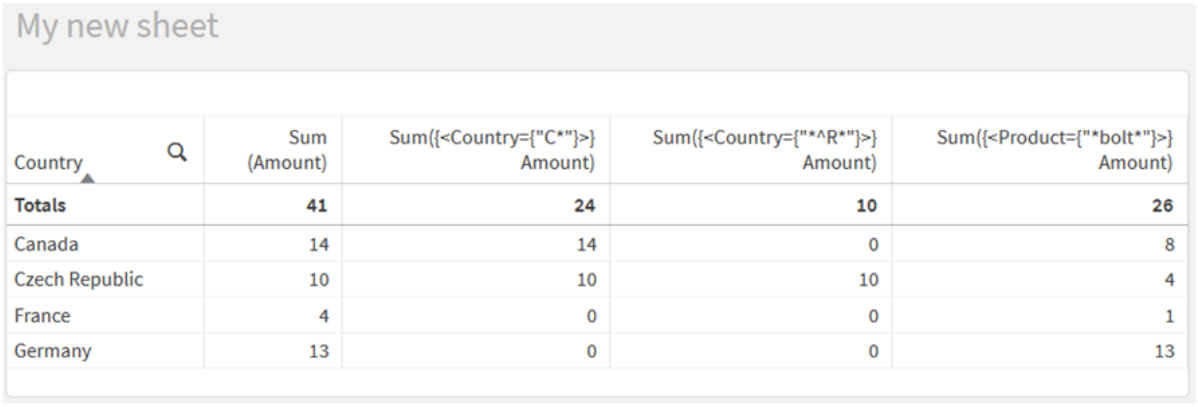
Tabelle – Modifikatoren für Auswahlformeln mit Textsuchen

Land	Sum (Amount)	Sum({<Country={"C*"}>} Amount)	Sum({<Country={"*^R*"}>} Amount)	Sum({<Product={"*bolt*"}>} Amount)
Summen	**41**	**24**	**10**	**26**
Kanada	14	14	0	8
Tschechische Republik	10	10	10	4
Frankreich	4	0	0	1
Deutschland	13	0	0	13

Erläuterung

- Dimension:
  - `Country`
- Kennzahl:
  - `Sum(Amount)`
    `Amount` ohne Auswahlformel summieren.
  - `Sum({<Country={"C*"}>}Amount)`
    `Amount` für alle Länder summieren, die mit `C` beginnen, beispielsweise `Canada` und `Czech Republic`.
  - `Sum({<Country={"*^R*"}>}Amount)`
    `Amount` für alle Länder summieren, die ein Wort enthalten, das mit `R` beginnt, z. B. `Czech Republic`.
  - `Sum({<Product={"*bolt*"}>}Amount)`
    `Amount` für alle Produkte summieren, die die Zeichenfolge `bolt` enthalten, wie `Bolt` und `Anchor bolt`.

Modifikatoren für Auswahlformeln mit Textsuchen



Country	Sum (Amount)	Sum({<Country={"C*"}>} Amount)	Sum({<Country={"*^R*"}>} Amount)	Sum({<Product={"*bolt*"}>} Amount)
Totals	**41**	**24**	**10**	**26**
Canada	14	14	0	8
Czech Republic	10	10	10	4
France	4	0	0	1
Germany	13	0	0	13

Beispiel 2: Diagrammformeln mit numerischen Suchen

Erstellen Sie eine Tabelle in einem Qlik Sense Arbeitsblatt mit den folgenden Diagrammformeln.

Tabelle – Modifikatoren für Auswahlformeln mit numerischen Suchen

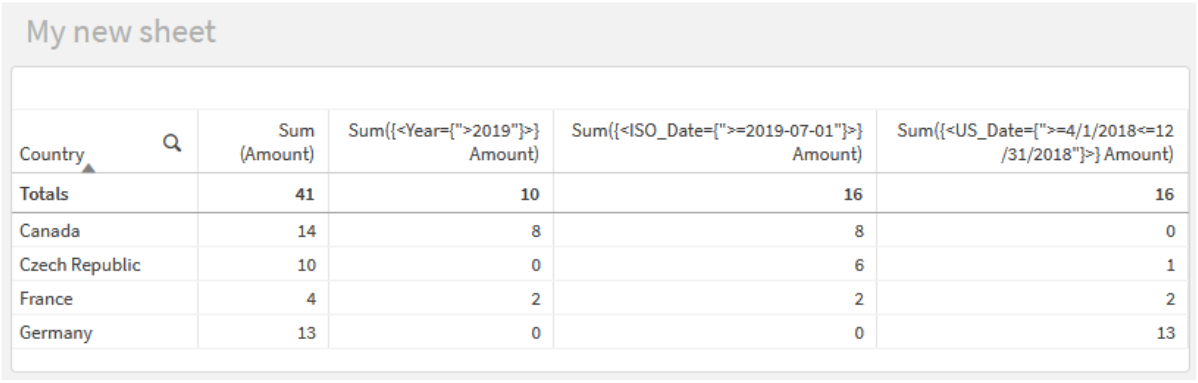
Land	**Sum (Amount)**	**Sum({<Year={">2019"}>} Amount)**	**Sum({<ISO_Date={">=2019-07-01"}>} Amount)**	**Sum({<US_Date={">=4/1/2018<=12/31/2018"}>} Amount)**
Summen	**41**	**10**	**16**	**16**
Kanada	14	8	8	0
Tschechische Republik	10	0	6	1
Frankreich	4	2	2	2
Deutschland	13	0	0	13

Erläuterung

- Dimension:
  - `Country`
- Kennzahl:
  - `Sum(Amount)`
    `Amount` ohne Auswahlformel summieren.
  - `Sum({<Year={">2019"}>}Amount)`
    `Amount` für alle Jahre nach 2019 summieren.
  - `Sum({<ISO_Date={">=2019-07-01"}>}Amount)`
    `Amount` für alle Tage am oder nach dem `2019-07-01` summieren. Das Format des Datums in der Suche muss dem Format des Feldes entsprechen.
  - `Sum({<US_Date={">=4/1/2018<=12/31/2018"}>}Amount)`

`Amount` für alle Tage vom `4/1/2018` bis zum `12/31/2018` summieren, einschließlich dem Start- und Enddatum. Das Format des Datums in der Suche muss dem Format des Feldes entsprechen.

Modifikatoren für Auswahlformeln mit numerischen Suchen



Beispiel 3: Diagrammformeln mit Formelsuchen

Erstellen Sie eine Tabelle in einem Qlik Sense Arbeitsblatt mit den folgenden Diagrammformeln.

Table - Set modifiers with expression searches

Country	Sum (Amount)	Sum({<Country={"=Sum (Amount)>10"}>} Amount)	Sum({<Country={"=Count(distinct Product)=1"}>} Amount)	Sum({<Product={"=Count (Amount)>3"}>} Amount)
Totals	**41**	**27**	**13**	**22**
Canada	14	14	0	8
Czech Republic	10	0	0	0
France	4	0	0	1
Germany	13	13	13	13

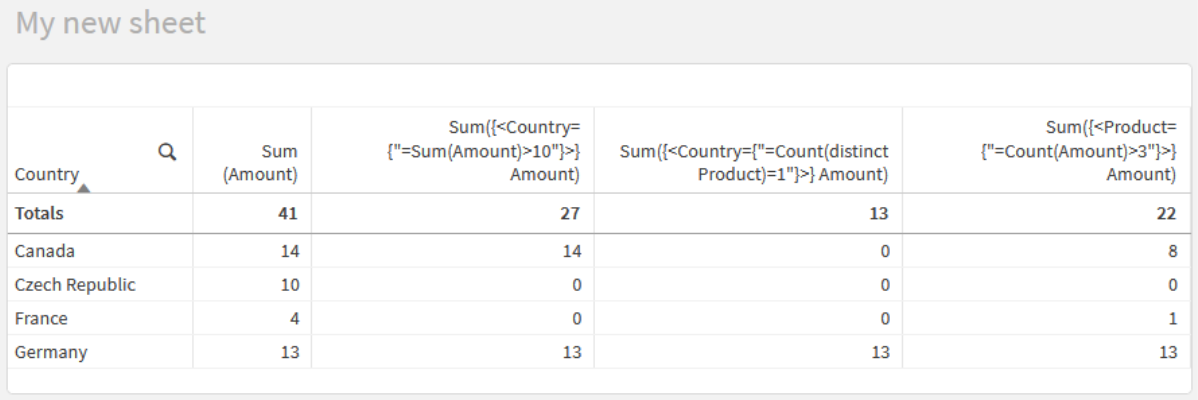
Erläuterung

- Dimension:
  - `Country`
- Kennzahl:
  - `Sum(Amount)`
    `Amount` ohne Auswahlformel summieren.
  - `Sum({<Country={"=Sum(Amount)>10"}>}Amount)`
    `Amount` für alle Länder summieren, die eine aggregierte Summe von `Amount` größer als `10` haben.
  - `Sum({<Country={"=Count(distinct Product)=1"}>}Amount)`

`Amount` für alle Länder summieren, die mit genau einem bestimmten Produkt verknüpft sind.

- `Sum({<Product={"=Count(Amount)>3"}>}Amount)`
  `Amount` für alle Länder summieren, die mehr als drei Transaktionen in den Daten enthalten.

Modifikatoren für Auswahlformeln mit Formelsuchen



Country	Sum (Amount)	Sum({<Country={"=Sum(Amount)>10"}>} Amount)	Sum({<Country={"=Count(distinct Product)=1"}>} Amount)	Sum({<Product={"=Count(Amount)>3"}>} Amount)
Totals	**41**	**27**	**13**	**22**
Canada	14	14	0	8
Czech Republic	10	0	0	0
France	4	0	0	1
Germany	13	13	13	13

Beispiele	Results
sum({$–1<Product = {“*Internal*”, “*Domestic*”}>} Sales)	Liefert den Umsatz für die aktuelle Auswahl, aber ohne alle Datensätze mit Produktnamen, die 'Internal' oder 'Domestic' enthalten.
sum({$<Customer = {“=Sum ({1<Year = {2007}>} Sales) > 1000000”}>} Sales)	Liefert den Umsatz für die aktuelle Auswahl, aber mit einer geänderten Auswahl im Feld 'Customer'. Nur Kunden, die im Jahr 2007 Umsätze > 1.000.000 hatten, werden berücksichtigt.

Modifikatoren mit Dollarzeichen

Dollarzeichen-Erweiterungen sind Konstrukte, die berechnet werden, bevor die Formel analysiert und ausgewertet wird. Das Ergebnis wird dann anstelle von `$(…)` in die Formel eingefügt. Die Berechnung der Formel erfolgt dann mit dem Ergebnis der Dollarzeichen-Erweiterung.

Der Formel-Editor zeigt eine Variablenvorschau mit Dollarzeichen-Erweiterung, damit Sie prüfen können, welches Ergebnis Ihre Dollarzeichen-Erweiterung zurückgibt.

Variablenvorschau mit Dollarzeichen-Erweiterung im Formel-Editor

Verwenden Sie Dollarzeichen-Erweiterungen, wenn Sie eine Berechnung innerhalb Ihres Elementsatzes verwenden möchten.

Wenn Sie beispielsweise nur das letzte mögliche Jahr betrachten möchten, können Sie folgende Konstruktion verwenden:

```
<Year = {$(=Max(Year))}>
```

`Max(Year)` wird zuerst berechnet, und das Ergebnis wird anstelle von `$(…)` in die Formel eingefügt.

Das Ergebnis nach der Dollarzeichenerweiterung ist eine Formel wie die folgende:

```
<Year = {2021}>
```

Die Formel innerhalb der Dollarzeichen-Erweiterung wird basierend auf der aktuellen Auswahl berechnet. Das bedeutet, dass sich eine Auswahl in einem anderen Feld auf das Ergebnis der Formel auswirkt.

Wenn die Berechnung unabhängig von der Auswahl sein soll, verwenden Sie eine Aggregierung mit Auswahlformeln innerhalb der Dollarzeichen-Erweiterung. Hier ein Beispiel:

```
<Year = {$(=Max({1} Year))}>
```

Zeichenfolgen

Wenn das Ergebnis der Dollarzeichen-Erweiterung eine Zeichenfolge sein soll, gelten normale Regeln für Anführungszeichen. Hier ein Beispiel:

```
<Country = {'$(=FirstSortedValue(Country,Date)'}>
```

Das Ergebnis nach der Dollarzeichenerweiterung ist eine Formel wie die folgende:

```
<Country = {'New Zealand'}>
```

Sie erhalten einen Syntaxfehler, wenn Sie die Anführungszeichen nicht verwenden.

Zahlen

Wenn das Ergebnis der Dollarzeichen-Erweiterung eine Zahl sein soll, vergewissern Sie sich, dass die Erweiterung die gleiche Formatierung wie das Feld enthält. Das bedeutet, dass die Formel manchmal von einer Formatierungsfunktion umschlossen sein muss.

Hier ein Beispiel:

```
<Amount = {$(=Num(Max(Amount), '###0.00'))}>
```

Das Ergebnis nach der Dollarzeichenerweiterung ist eine Formel wie die folgende:

```
<Amount = {12362.00}>
```

Verwenden Sie ein Hash, um zu erzwingen, dass immer der Dezimalpunkt und kein Tausendertrennzeichen verwendet wird. Hier ein Beispiel:

```
<Amount = {$(#=Max(Amount))}>
```

Datumsangaben

Wenn das Ergebnis der Dollarzeichen-Erweiterung ein Datum sein soll, vergewissern Sie sich, dass die Erweiterung die korrekte Formatierung aufweist. Das bedeutet, dass die Formel manchmal von einer Formatierungsfunktion umschlossen sein muss.

Hier ein Beispiel:

```
<Date = {'$(=Date(Max(Date)))'}>
```

Das Ergebnis nach der Dollarzeichenerweiterung ist eine Formel wie die folgende:

```
<Date = {'12/31/2015'}>
```

Wie bei Zeichenfolgen müssen Sie die richtigen Anführungszeichen verwenden.

Ein häufiger Anwendungsfall ist, dass die Berechnung auf den letzten Monat oder das letzte Jahr beschränkt sein soll. Dann können Sie eine numerische Suche in Kombination mit der Funktion `AddMonths()` verwenden.

Hier ein Beispiel:

```
<Date = {">=$(=AddMonths(Today(),-1))"}>
```

Das Ergebnis nach der Dollarzeichenerweiterung ist eine Formel wie die folgende:

```
<Date = {">=9/31/2021"}>
```

Damit werden alle Ereignisse herausgesucht, die im letzten Monat eingetreten sind.

Beispiel: Diagrammformeln für Modifikatoren für Auswahlformen mit Dollarzeichen-Erweiterungen

Beispiel – Diagrammformeln

Ladeskript

Laden Sie die folgenden Daten als Inline-Ladevorgang in den Dateneditor, um die folgenden Diagrammformelbeispiele zu erstellen.

```
Let vToday = Today();
MyTable:
Load
```

```
Year(Date) as Year,
Date#(Date,'YYYY-MM-DD') as ISO_Date,
Date(Date#(Date,'YYYY-MM-DD'),'M/D/YYYY') as US_Date,
Country, Product, Amount
Inline
[Date, Country, Product, Amount
2018-02-20, Canada, Washer, 6
2018-07-08, Germany, Anchor bolt, 10
2018-07-14, Germany, Anchor bolt, 3
2018-08-31, France, Nut, 2
2018-09-02, Czech Republic, Bolt, 1
2019-02-11, Czech Republic, Bolt, 3
2019-07-31, Czech Republic, Washer, 6
2020-03-13, France, Anchor bolt, 1
2020-07-12, Canada, Anchor bolt, 8
2021-10-15, France, Washer, 1];
```

Diagrammformeln mit Dollarzeichen-Erweiterungen

Erstellen Sie eine Tabelle in einem Qlik Sense Arbeitsblatt mit den folgenden Diagrammformeln.

Tabelle – Modifikatoren für Auswahlformeln zum Aufrufen von Variablen (Dollarzeichen-Erweiterungen)

Land	**Sum (Amount)**	**Sum({<US_Date={'$(vToday)'}>} Amount)**	**Sum({<ISO_Date={"$(=Date(Min(ISO_Date),'YYYY-MM-DD'))"}>} Amount)**	**Sum({<US_Date={">=$(=AddYears(Max(US_Date),-1))"}>} Amount)**
Summen	**41**	**1**	**6**	**1**
Kanada	14	0	6	0
Tschechische Republik	10	0	0	0
Frankreich	4	1	0	1
Deutschland	13	0	0	0

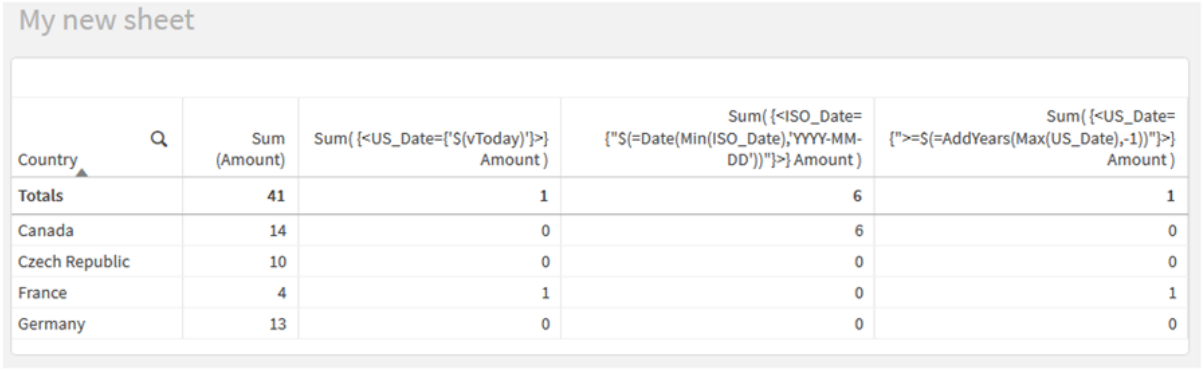
Erläuterung

- Dimension:
  - `Country`
- Kennzahl:
  - `Sum(Amount)`
    `Amount` ohne Auswahlformel summieren.
  - `Sum({<US_Date={'$(vToday)'}>}Amount)`
    `Amount` für alle Datensätze summieren, bei denen das `US_Date` dasselbe ist wie in der Variablen `vToday`.
  - `Sum({<ISO_Date={"$(=Date(Min(ISO_Date),'YYYY-MM-DD'))"}>}Amount)`

`Amount` für alle Datensätze summieren, bei denen das `ISO_Date` dasselbe wie das erste (früheste) mögliche `ISO_Date` ist. Die Funktion `Date()` wird benötigt, um sicherzustellen, dass das Format des Datums mit dem des Felds übereinstimmt.

- `Sum({<US_Date={">=$(=AddYears(Max(US_Date),-1))"}>}Amount)`
  `Amount` für alle Datensätze summieren, die einen `US_Date` nach dem oder am Datum ein Jahr vor dem letzten (spätesten) möglichen `US_Date` haben. Die Funktion `AddYears()` gibt ein Datum in dem Format zurück, das von der Variablen `DateFormat` angegeben wird, und dieses muss dem Format des Felds `US_Date` entsprechen.

Modifikatoren mit Dollarzeichen



Country	Sum (Amount)	Sum({<US_Date={'$(vToday)'}>} Amount)	Sum({<ISO_Date={"$(=Date(Min(ISO_Date),'YYYY-MM-DD'))"}>} Amount)	Sum({<US_Date={">=$(=AddYears(Max(US_Date),-1))"}>} Amount)
Totals	**41**	**1**	**6**	**1**
Canada	14	0	6	0
Czech Republic	10	0	0	0
France	4	1	0	1
Germany	13	0	0	0

Beispiele	Ergebnisse
sum({$<Year = {$(#vLastYear)}>} Sales)	Liefert den Umsatz für das Vorjahr in Bezug auf die aktuelle Auswahl. Hier wird eine Dollarzeichenformel mit der Variablen vLastYear mit dem relevanten Jahr verwendet.
sum({$<Year = {$(#=Only(Year)-1)}>} Sales)	Liefert den Umsatz für das Vorjahr in Bezug auf die aktuelle Auswahl. Das Vorjahr wird durch eine Dollarzeichenformel berechnet.

Modifikatoren mit Operatoren

Operatoren für Auswahlformeln werden zum Einschließen, Ausschließen und Überschneiden verschiedener Elementsätze verwendet. Sie kombinieren verschiedene Methoden zum Definieren von Elementsätzen.

Die Operatoren sind die gleichen wie diejenigen, die für Identifikatoren für Auswahlformeln verwendet werden.

Operatoren

Operator	Beschreibung
+	Vereinigung. Diese zweiwertige Operation liefert einen Satz aus Datensätzen oder Elementen, die zu mindestens einem der beiden Operanden gehören.

Operator	Beschreibung
-	Ausschluss. Diese zweiwertige Operation liefert einen Satz aus Datensätzen oder Elemente, die zum ersten, nicht aber zum zweiten Operanden gehören. Als einwertiger Operator liefert er die Umkehrung der momentanen Auswahl.
*	Schnittmenge. Diese zweiwertige Operation liefert einen Satz aus Datensätzen oder Elemente, die zu beiden Operanden gehören.
/	Symmetrische Differenz (XOR). Diese zweiwertige Operation liefert einen Satz aus Datensätzen oder Elementen, die zu einem der Operanden, aber nicht zu beiden gehören.

Beispielsweise definieren die folgenden beiden Modifikatoren den gleichen Satz Feldwerte:

- `<Year = {1997, "20*"}>`
- `<Year = {1997} + {"20*"}>`

Beide Formeln wählen 1997 und die Jahre, die mit 20 beginnen. Es handelt sich also um eine Verbindung der beiden Bedingungen.

Operatoren für Auswahlformeln lassen auch komplexere Definitionen zu. Hier ein Beispiel:

```
<Year = {1997, "20*"} - {2000}>
```

Mit dieser Formel werden die gleichen Jahre wie oben ausgewählt, aber zusätzlich das Jahr 2000 ausgeschlossen.

.

Beispiele: Diagrammformeln für Modifikatoren für Auswahlformen mit Operatoren

Beispiele – Diagrammformeln

Ladeskript

Laden Sie die folgenden Daten als Inline-Ladevorgang in den Dateneditor, um die folgenden Diagrammformelbeispiele zu erstellen.

```
MyTable:
Load
Year(Date) as Year,
Date#(Date,'YYYY-MM-DD') as ISO_Date,
Date(Date#(Date,'YYYY-MM-DD'),'M/D/YYYY') as US_Date,
Country, Product, Amount
Inline
[Date, Country, Product, Amount
2018-02-20, Canada, Washer, 6
2018-07-08, Germany, Anchor bolt, 10
2018-07-14, Germany, Anchor bolt, 3
2018-08-31, France, Nut, 2
2018-09-02, Czech Republic, Bolt, 1
2019-02-11, Czech Republic, Bolt, 3
```

```
2019-07-31, Czech Republic, Washer, 6
2020-03-13, France, Anchor bolt, 1
2020-07-12, Canada, Anchor bolt, 8
2020-09-16, France, Washer, 1];
```

Diagrammformeln

Erstellen Sie eine Tabelle in einem Qlik Sense Arbeitsblatt mit den folgenden Diagrammformeln.

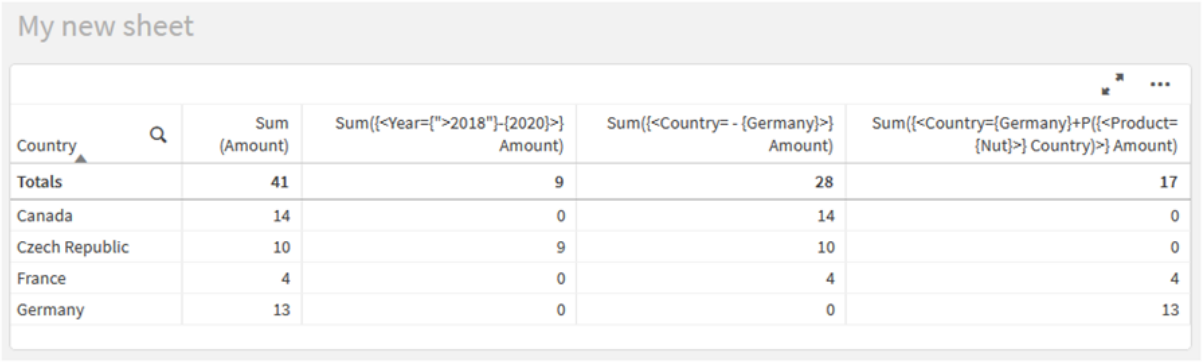
Tabelle – Modifikatoren für Auswahlformeln mit Operatoren

Land	Sum (Amount)	Sum({<Year={">2018"}-{2020}>} Amount)	Sum ({<Country=-{Germany}>} Amount)	Sum({<Country={Germany}+P({<Product={Nut}>}Country)>} Amount)
Summen	**41**	**9**	**28**	**17**
Kanada	14	0	14	0
Tschechische Republik	10	9	10	0
Frankreich	4	0	4	4
Deutschland	13	0	0	13

Erläuterung

- Dimension:
  - `Country`
- Kennzahl:
  - `Sum(Amount)`
    `Amount` ohne Auswahlformel summieren.
  - `Sum({<Year={">2018"}-{2020}>}Amount)`
    `Amount` für alle Jahre nach 2018 außer 2020 summieren.
  - `Sum({<Country=-{Germany}>}Amount)`
    `Amount` für alle Länder außer `Germany` summieren. Beachten Sie den unären Ausschlussoperator.
  - `Sum({<Country={Germany}+P({<Product={Nut}>}Country)>}Amount)`
    `Amount` für `Germany` und für alle Länder summieren, die nicht mit dem Produkt `Nut` verknüpft sind.

Modifikatoren mit Operatoren


My new sheet

Country	Sum (Amount)	Sum({<Year={">2018"}-{2020}>} Amount)	Sum({<Country= - {Germany}>} Amount)	Sum({<Country={Germany}+P({<Product={Nut}>} Country)>} Amount)
Totals	**41**	**9**	**28**	**17**
Canada	14	0	14	0
Czech Republic	10	9	10	0
France	4	0	4	4
Germany	13	0	0	13

Beispiele	**Ergebnisse**
sum({$<Product = Product + {OurProduct1} – {OurProduct2} >} Sales)	Liefert den Umsatz für die aktuelle Auswahl, wobei der bestehenden Auswahl im Feld das Produkt “OurProduct1” hinzugefügt und das Produkt “OurProduct2” daraus entfernt wird.
sum({$<Year = Year + ({“20*”,1997} – {2000}) >} Sales)	Liefert den Umsatz für die aktuelle Auswahl, aber mit zusätzlichen Auswahlen im Feld “Year”: 1997 und alle Jahre, die mit 20 beginnen, werden der Auswahl hinzugefügt, 2000 jedoch nicht. Bitte beachten Sie Folgendes: Ist 2000 Teil der bestehenden Auswahl, wird es auch nach der Modifikation Teil der Auswahl sein.
sum({$<Year = (Year + {“20*”,1997}) – {2000} >} Sales)	Diese Formel liefert beinahe dasselbe Ergebnis wie das vorangehende Beispiel, aber hier ist das Jahr 2000 ausgeschlossen, selbst wenn es Teil der bestehenden Auswahl ist. Im Beispiel wird die Bedeutung der Klammern zum Festlegen einer Rangfolge deutlich.
sum({$<Year = {“*”} – {2000}, Product = {“*bearing*”} >} Sales)	Liefert den Umsatz für die aktuelle Auswahl, aber mit einer neuen Auswahl in “Year”: für alle Jahre mit Ausnahme des Jahres 2000 und beschränkt auf alle Produkte, die den String 'bearing' enthalten.

Modifikatoren für Auswahlformeln mit impliziten Operatoren für Auswahlformeln

Standardmäßig werden Auswahlen in einem Modifikator für Auswahlformeln mit einem Gleichheitszeichen geschrieben. Hier ein Beispiel:

```
Year = {">2015"}
```

Die Formel rechts neben dem Gleichheitszeichen im Modifikator für Auswahlformeln wird als Elementsatz bezeichnet. Er definiert einen Satz distinkter Feldwerte, in anderen Worten eine Auswahl.

Diese Schreibweise definiert eine neue Auswahl. Die aktuelle Auswahl im Feld wird ignoriert. Wenn also der Identifikator für Auswahlformeln eine Auswahl in diesem Feld enthält, wird die alte Auswahl durch die Auswahl im Elementsatz ersetzt.

Wenn Ihre Auswahl auf der aktuellen Auswahl im Feld basieren soll, müssen Sie eine andere Formel verwenden.

Wenn Sie beispielsweise die alte Auswahl beibehalten und die Anforderung, dass das Jahr nach 2015 liegt, hinzufügen möchten, schreiben Sie Folgendes:

```
Year = Year * {">2015"}
```

Der Asterisk ist ein Operator für Auswahlformeln, der eine Schnittmenge definiert. Daher erhalten Sie die Schnittmenge zwischen der aktuellen Auswahl in `Year` und der zusätzlichen Anforderung, dass das Jahr nach 2015 liegt. Alternativ können Sie dies wie folgt schreiben:

```
Year *= {">2015"}
```

Der Zuweisungsoperator (*=) definiert also implizit eine Schnittmenge.

Auf ähnliche Weise können implizite Verbindungen, Ausschlüsse und symmetrische Differenzen mit folgenden Zeichen definiert werden: +=, –=, /=

Beispiele: Diagrammformeln für Modifikatoren für Auswahlformen mit impliziten Operatoren für Auswahlformeln

Beispiele – Diagrammformeln

Ladeskript

Laden Sie die folgenden Daten als Inline-Ladevorgang in den Dateneditor, um die folgenden Diagrammformelbeispiele zu erstellen.

```
MyTable:
Load
Year(Date) as Year,
Date#(Date,'YYYY-MM-DD') as ISO_Date,
Date(Date#(Date,'YYYY-MM-DD'),'M/D/YYYY') as US_Date,
Country, Product, Amount
Inline
[Date, Country, Product, Amount
2018-02-20, Canada, Washer, 6
2018-07-08, Germany, Anchor bolt, 10
2018-07-14, Germany, Anchor bolt, 3
2018-08-31, France, Nut, 2
2018-09-02, Czech Republic, Bolt, 1
2019-02-11, Czech Republic, Bolt, 3
2019-07-31, Czech Republic, Washer, 6
2020-03-13, France, Anchor bolt, 1
2020-07-12, Canada, Anchor bolt, 8
2020-09-16, France, Washer, 1];
```

Diagrammformeln mit impliziten Operatoren für Auswahlformeln

Erstellen Sie eine Tabelle in einem Qlik Sense Arbeitsblatt mit den folgenden Diagrammformeln.

`Canada` und `Czech Republic` aus einer Länderliste auswählen.

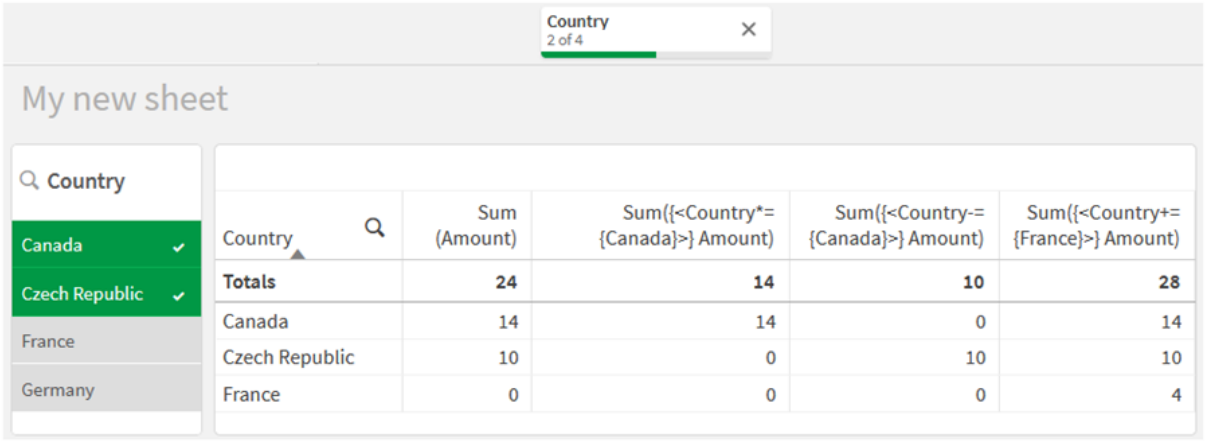
Tabelle – Diagrammformeln mit impliziten Operatoren für Auswahlformeln

Land	Sum (Amount)	Sum({<Country*= {Canada}>} Amount)	Sum({<Country-= {Canada}>} Amount)	Sum({<Country+= {France}>} Amount)
Summen	**24**	**14**	**10**	**28**
Kanada	14	14	0	14
Tschechische Republik	10	0	10	10
Frankreich	0	0	0	4

Erläuterung

- Dimension:
  - `Country`
- Kennzahl:
  - `Sum(Amount)`
    `Amount` für die aktuelle Auswahl summieren. Beachten Sie, dass nur `Canada` und `Czech Republic` Nicht-Null-Werte haben.
  - `Sum({<Country*={Canada}>}Amount)`
    `Amount` für die aktuelle Auswahl summieren, überschnitten mit der Anforderung, dass das L`Country` `Canada` sein muss. Wenn `Canada` nicht Teil der Benutzerauswahl ist, gibt die Auswahlformel einen leeren Satz zurück, und die Spalte enthält `0` in allen Zeilen.
  - `Sum({<Country-={Canada}>}Amount)`
    `Amount` für die aktuelle Auswahl summieren, aber zuerst `Canada` aus der Auswahl `Country` ausschließen. Wenn `Canada` nicht Teil der Benutzerauswahl ist, ändert die Auswahlformel keine Zahlen.
  - `Sum({<Country+={France}>}Amount)`
    `Amount` für die aktuelle Auswahl summieren, aber zuerst `France` zur Auswahl `Country` hinzufügen. Wenn `France` nicht bereits Teil der Benutzerauswahl ist, ändert die Auswahlformel keine Zahlen.

Modifikatoren für Auswahlformeln mit impliziten Operatoren für Auswahlformeln.



Beispiele	Ergebnisse
sum({$<Product += {OurProduct1, OurProduct2} >} Sales)	Liefert den Umsatz für die aktuelle Auswahl, wobei der bestehenden Auswahl im Feld die Produkte 'OurProduct1' und 'OurProduct2' hinzugefügt werden.
sum({$<Year += {“20*”,1997} – {2000} >} Sales)	Liefert den Umsatz für die aktuelle Auswahl, wobei der bestehenden Auswahl einige Jahre hinzugefügt werden: 1997 und alle Jahre, die mit 20 beginnen, werden der Auswahl hinzugefügt, 2000 jedoch nicht. Bitte beachten Sie Folgendes: Ist 2000 Teil der bestehenden Auswahl, wird es auch nach der Modifikation Teil der Auswahl sein. Entspricht `<Year=Year + ({“20*”,1997}–{2000})>`.
sum({$<Product *= {OurProduct1} >} Sales)	Liefert den Umsatz für die aktuelle Auswahl, aber nur für die Schnittmenge der aktuell ausgewählten Produkte und des Produkts OurProduct1.

Modifikatoren für Auswahlformeln mit Funktionen für Auswahlformeln

Manchmal müssen Sie eine Reihe von Feldwerten mit einer verschachtelten Auswahlfunktion definieren. Beispiel: Sie möchten alle Kunden auswählen, die ein spezifisches Produkt gekauft haben, ohne das Produkt auszuwählen.

In solchen Fällen verwenden Sie die Elementsatzfunktionen `P()` und `E()`. Diese geben die Elementsätze möglicher Werte bzw. ausgeschlossener Werte eines Feldes zurück. Innerhalb der Klammern können Sie das betreffende Feld und eine Auswahlformel, die den Umfang definiert, angeben. Hier ein Beispiel:

```
P({1<Year = {2021}>} Customer)
```

Damit wird der Satz Kunden zurückgegeben, die im Jahr 2021 Transaktionen hatten. Dies können Sie dann in einem Modifikator für Auswahlformeln verwenden. Hier ein Beispiel:

```
Sum({<Customer = P({1<Year = {2021}>} Customer)>} Amount)
```

Mit dieser Auswahlformel werden diese Kunden ausgewählt, aber die Auswahl wird nicht auf 2021 beschränkt.

In anderen Ausdrücken können diese Funktionen nicht eingesetzt werden:

Zudem können nur natürliche Sätze innerhalb der Elementsatzfunktionen verwendet werden. Ein natürlicher Satz bezeichnet einen Datensatz, der durch einen einfachen Auswahlvorgang definiert werden kann.

Ein durch `{1-$}` angegebener Satz kann beispielsweise nicht immer durch eine Auswahl definiert werden und ist somit kein natürlicher Satz. In nicht natürlichen Sätzen geben diese Funktionen unerwartete Ergebnisse zurück.

Beispiele: Diagrammformeln für Modifikatoren für Auswahlformeln mit Funktionen für Auswahlformeln

Beispiele – Diagrammformeln

Ladeskript

Laden Sie die folgenden Daten als Inline-Ladevorgang in den Dateneditor, um die folgenden Diagrammformelbeispiele zu erstellen.

```
MyTable:
Load
Year(Date) as Year,
Date#(Date,'YYYY-MM-DD') as ISO_Date,
Date(Date#(Date,'YYYY-MM-DD'),'M/D/YYYY') as US_Date,
Country, Product, Amount
Inline
[Date, Country, Product, Amount
2018-02-20, Canada, Washer, 6
2018-07-08, Germany, Anchor bolt, 10
2018-07-14, Germany, Anchor bolt, 3
2018-08-31, France, Nut, 2
2018-09-02, Czech Republic, Bolt, 1
2019-02-11, Czech Republic, Bolt, 3
2019-07-31, Czech Republic, Washer, 6
2020-03-13, France, Anchor bolt, 1
2020-07-12, Canada, Anchor bolt, 8
2020-09-16, France, Washer, 1];
```

Diagrammformeln

Erstellen Sie eine Tabelle in einem Qlik Sense Arbeitsblatt mit den folgenden Diagrammformeln.

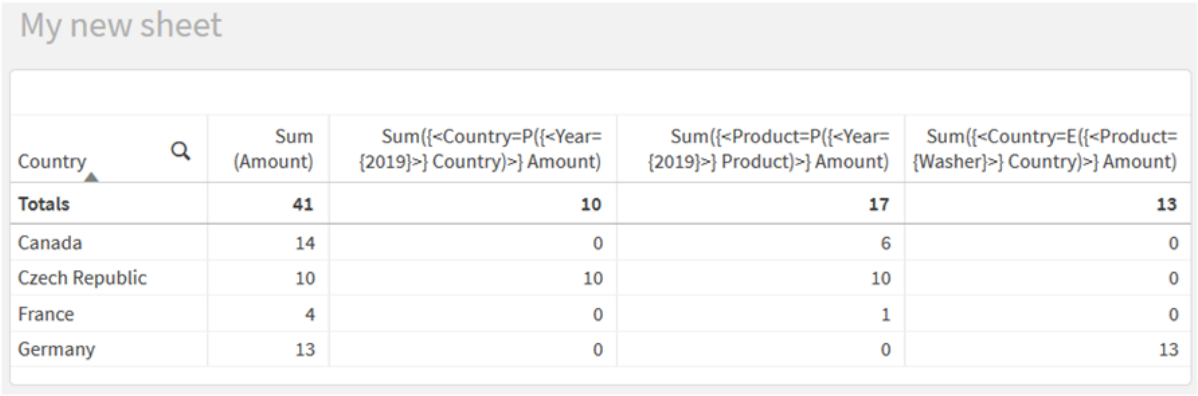
Tabelle – Modifikatoren für Auswahlformeln mit Funktionen für Auswahlformeln

Land	Sum (Amount)	Sum({<Country=P({<Year={2019}>}Country)>} Amount)	Sum({<Product=P({<Year={2019}>}Product)>} Amount)	Sum({<Country=E({<Product={Washer}>}Country)>} Amount)
Summen	**41**	**10**	**17**	**13**
Kanada	14	0	6	0
Tschechische Republik	10	10	10	0
Frankreich	4	0	1	0
Deutschland	13	0	0	13

Erläuterung

- Dimension:
  - `Country`
- Kennzahl:
  - `Sum(Amount)`
    `Amount` ohne Auswahlformel summieren.
  - `Sum({<Country=P({<Year={2019}>} Country)>} Amount)`
    `Amount` für alle Länder summieren, die mit dem Jahr 2019 verknüpft sind. Damit wird aber die Berechnung nicht auf 2019 beschränkt.
  - `Sum({<Product=P({<Year={2019}>} Product)>} Amount)`
    `Amount` für die Produkte summieren, die mit dem Jahr 2019 verknüpft sind. Damit wird aber die Berechnung nicht auf 2019 beschränkt.
  - `Sum({<Country=E({<Product={Washer}>} Country)>} Amount)`
    `Amount` für alle Länder summieren, die nicht mit dem Produkt `Washer` verknüpft sind.

Modifikatoren für Auswahlformeln mit Funktionen für Auswahlformeln



Beispiele	Results
sum({$<Customer = P({1<Product= {'Shoe'}>} Customer)>} Sales)	Liefert den Umsatz für die aktuelle Auswahl, jedoch nur für die Kunden, die das Produkt 'Shoe' bereits gekauft haben. Die Statusfunktion P() liefert die wählbaren Werte des Felds "Kunde". Diese sind bestimmt durch die Auswahl des Werts 'Shoe' im Feld Product.
sum({$<Customer = P({1<Product= {'Shoe'}>})>} Sales)	Wie oben. Enthält das Argument der Statusfunktion keinen Feldnamen, liefert die Funktion alle wählbaren Werte der in der Auswahlformel angegebenen Felder.
sum({$<Customer = P({1<Product= {'Shoe'}>} Supplier)>} Sales)	Liefert den Umsatz für die aktuelle Auswahl, jedoch nur für die Kunden, die das Produkt 'Shoe' bereits geliefert haben. In diesem Fall ist der Kunde also auch ein Lieferant. Die Statusfunktion P() liefert die wählbaren Werte des Felds "Lieferant". Diese sind bestimmt durch die Auswahl des Werts 'Shoe' im Feld Product. Diese Lieferanten werden im Feld Customer ausgewählt.
sum({$<Customer = E({1<Product= {'Shoe'}>})>} Sales)	Liefert den Umsatz für die aktuelle Auswahl, jedoch nur für die Kunden, die das Produkt 'Shoe' noch nie gekauft haben. Die Statusfunktion E() liefert die ausgeschlossenen Werte des Felds "Kunde". Diese sind bestimmt durch die Auswahl des Werts 'Shoe' im Feld Product.

Innere und äußere Auswahlformeln

Auswahlformeln können inner- und außerhalb von Aggregierungsfunktionen verwendet werden und sind in geschweifte Klammern eingeschlossen.

Wenn Sie eine Auswahlformel innerhalb einer Aggregierungsfunktion verwenden, kann dies so aussehen:

Beispiel: Innere Auswahlformel

```
Sum( {$<Year={2021}>} Sales )
```

Verwenden Sie eine Auswahlformel außerhalb der Aggregierungsfunktion, wenn Sie Formeln mit mehreren Aggregierungen haben und vermeiden möchten, die gleiche Auswahlformel in jede Aggregierungsfunktion zu schreiben.

Wenn Sie eine äußere Auswahlformel verwenden, muss sie am Beginn des Geltungsbereichs platziert werden.

Beispiel: Äußere Auswahlformel

```
{<Year={2021}>} Sum(Sales) / Count(distinct Customer)
```

Wenn Sie eine Auswahlformel außerhalb der Aggregierungsfunktion verwenden, können Sie sie auch auf vorhandene Master-Kennzahlen anwenden.

Beispiel: Auf Master-Kennzahl angewendete äußere Auswahlformel

```
{<Year={2021}>} [Master Measure]
```

Eine Auswahlformel, die außerhalb von Aggregierungsfunktionen verwendet wird, wirkt sich auf die ganze Formel aus, es sei denn, sie wird in Klammern eingeschlossen. In diesem Fall definieren die Klammern den Geltungsbereich. In dem folgenden Beispiel für einen lexikalischen Geltungsbereich wird die Auswahlformel nur auf die Aggregierung innerhalb der Klammern angewandt.

Beispiel: Lexikalischer Geltungsbereich

```
( {<Year={2021}>} Sum(Amount) / Count(distinct Customer) ) – Avg(CustomerSales)
```

Regeln

Lexikalischer Geltungsbereich

Die Auswahlformel wirkt sich auf die ganze Formel aus, es sei denn, sie ist in Klammern eingeschlossen. In diesem Fall definieren die Klammern den lexikalischen Geltungsbereich.

Position

Die Auswahlformel muss am Anfang des lexikalischen Geltungsbereichs platziert werden.

Kontext

Der Kontext ist die Auswahl, die für die Formel relevant ist. Traditionell war der Kontext immer der Standardzustand der aktuellen Auswahl. Wenn ein Objekt jedoch auf alternative Zustände festgelegt ist, ist der Kontext der alternative Zustand der aktuellen Auswahl.

Sie können einen Kontext auch in Form einer äußeren Auswahlformel definieren.

Erben

Innere Auswahlformeln haben Vorrang vor äußeren Auswahlformeln. Wenn die innere Auswahlformel einen Identifikator für Auswahlformeln enthält, ersetzt sie den Kontext. Andernfalls werden der Kontext und die Auswahlformel zusammengeführt.

- `{$<SetExpression>}`: Überschreibt die äußere Auswahlformel.
- `{<SetExpression>}`: Wird mit der äußeren Auswahlformel zusammengeführt.

Elementsatzzuweisung

Die Elementsatzzuweisung bestimmt, wie die beiden Auswahlen zusammengeführt werden. Wenn ein normales Gleichheitszeichen verwendet wird, hat die Auswahl in der inneren Auswahlformel Vorrang. Andernfalls wird der implizite Operator für Auswahlformeln verwendet.

- `{<Field={value}>}`: Diese innere Auswahl ersetzt alle äußeren Auswahlen in "`Field`".
- `{<Field+={value}>}`: Diese innere Auswahl wird mit der äußeren Auswahl in "`Field`" zusammengeführt, wobei der Verbindungsoperator verwendet wird.
- `{<Field*={value}>}`: Diese innere Auswahl wird mit der äußeren Auswahl in "`Field`" zusammengeführt, wobei der Schnittoperator verwendet wird.

Erben in mehreren Schritten

Das Erben kann in mehreren Schritten stattfinden. Beispiele:

- Aktuelle Auswahl → `Sum(Amount)`
  Die Aggregierungsfunktion verwendet den Kontext, in diesem Fall die aktuelle Auswahl.
- Aktuelle Auswahl → `{<Set1>} Sum(Amount)`
  `Set1` erbt von der aktuellen Auswahl, und das Ergebnis ist der Kontext für die Aggregierungsfunktion.
- Aktuelle Auswahl → `{<Set1>} ({<Set2>} Sum(Amount))`
  `Set2` erbt von `Set1`, der wiederum von der aktuellen Auswahl erbt, und das Ergebnis ist der Kontext für die Aggregierungsfunktion.

Die Funktion Aggr()

Die Funktion `Aggr()` erstellt eine verschachtelte Aggregierung, die zwei unabhängige Aggregierungen enthält. Im Beispiel unten wird `Count()` für jeden Wert von `Dim` berechnet, und das sich ergebende Array wird anhand der Funktion `Sum()` aggregiert.

Beispiel:

`Sum(Aggr(Count(X),Dim))`
`Count()` ist die innere Aggregierung und `Sum()` ist die äußere Aggregierung.

- Die innere Aggregierung erbt keinen Kontext von der äußeren Aggregierung.
- Die innere Aggregierung erbt den Kontext von der Funktion `Aggr()`, die eine Auswahlformel enthalten kann.
- Sowohl die Funktion `Aggr()` als auch die äußere Aggregierungsfunktion erben den Kontext von einer äußeren Auswahlformel.

Tutorial – Erstellen einer Auswahlformel

Sie können Auswahlformeln in Qlik Sense so zusammenstellen, dass Datenanalyse unterstützt wird. In diesem Kontext wird die Analyse oft als Aggregierung mit Auswahlformeln bezeichnet. Die Aggregierung mit Auswahlformeln bietet eine Möglichkeit, einen Umfang zu definieren, der sich vom durch die aktuellen Auswahlen in einer App definierten Satz an Datensätzen unterscheidet.

Lerninhalte

In diesem Tutorial werden die Daten und Diagrammformeln gezeigt, mit denen Modifikatoren, Identifikatoren und Operatoren für Auswahlformeln erstellt werden.

Für wen wird dieses Tutorial empfohlen

Dieses Tutorial richtet sich an App-Entwickler, die mit dem Skript-Editor und mit Diagrammformeln vertraut sind.

Vorbereitungen vor dem Beginn

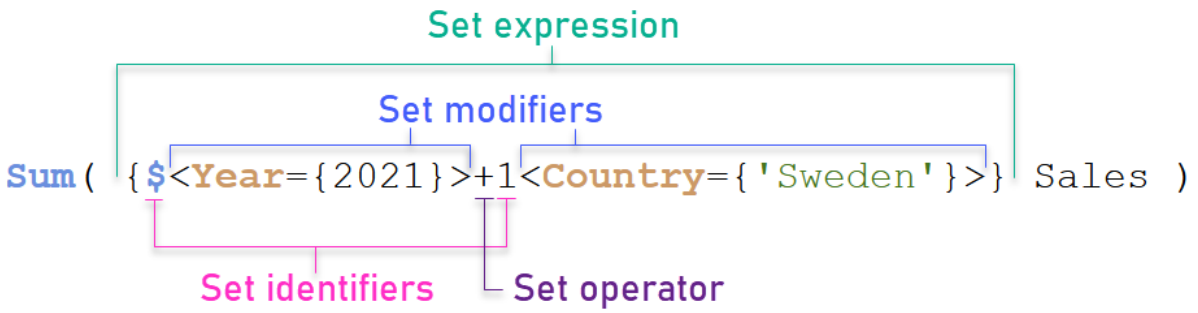
Sie benötigen die Zuteilung eines Qlik Sense Enterprise Professional-Zugriffs, um Daten laden und Apps erstellen zu können.

- Aggregierung mit Auswahlformeln Teil 1: Einführung für Anfänger
- Aggregierung mit Auswahlformeln Teil 2

Elemente in einer Auswahlformel

Auswahlformeln sind in eine Aggregierungsfunktion eingeschlossen, wie `Sum()`, `Max()`, `Min()`, `Avg()` oder `Count()`. Auswahlformeln sind aus Blöcken aufgebaut, die als Elemente bezeichnet werden. Diese Elemente sind Modifikatoren, Identifikatoren und Operatoren für Auswahlformeln.

Elemente in einer Auswahlformel



Die obige Auswahlformel wird beispielsweise anhand der Aggregierung `Sum(Sales)` erstellt. Die Auswahlformel ist in die äußeren geschweiften Klammern eingeschlossen: `{ }`

Der erste Operand in der Formel ist: `$<Year={2021}>`

Dieser Operand gibt den Umsatz für das Jahr `2021` für die aktuelle Auswahl zurück. Der Modifikator `<Year={2021}>` enthält die Auswahl des Jahres 2021. Der Identifikator für Auswahlformeln `$` gibt an, dass die Auswahlformel auf der aktuellen Auswahl basiert.

Der zweite Operand in der Formel ist: `1<Country={'Sweden'}>`

Dieser Operand gibt Sales für Sweden zurück. Der Modifikator `<Country={'Sweden'}>` enthält die Auswahl des Landes Sweden. Der Identifikator für Auswahlformeln `1` gibt an, dass die in der App getroffenen Auswahlen ignoriert werden.

Schließlich gibt der Operator für Auswahlformeln `+` an, dass die Formel einen Satz bestehend aus den Datensätzen zurückgibt, die zu einem beliebigen der beiden Operanden gehören.

Erstellen einer Auswahlformel – Tutorial

Befolgen Sie das folgende Verfahren, um die in diesem Tutorial gezeigten Auswahlformeln zu erstellen.

Eine neue App erstellen und Daten laden

Gehen Sie folgendermaßen vor:

1. Erstellen Sie eine neue App.
2. Klicken Sie auf **Skript-Editor**. Klicken Sie alternativ auf **Vorbereiten** > **Dateneditor** in der Navigationsleiste.
3. Erstellen Sie einen neuen Abschnitt im **Dateneditor**.

4. Kopieren Sie die folgenden Daten und fügen Sie sie in den neuen Abschnitt ein: *Tutorial-Daten für Auswahlformeln (page 331)*
5. Klicken Sie auf **Daten laden**. Die Daten werden als Inline-Ladevorgang geladen.

Erstellen von Auswahlformeln mit Modifikatoren

Der Modifikator für Auswahlformeln besteht aus mindestens einem Feldnamen, jeweils gefolgt von einer Auswahl von Feldwerten. Der Modifikator ist in spitze Klammern eingeschlossen. Beispielsweise gilt in dieser Auswahlformel:

```
Sum ( {<Year = {2015}>} Sales )
```

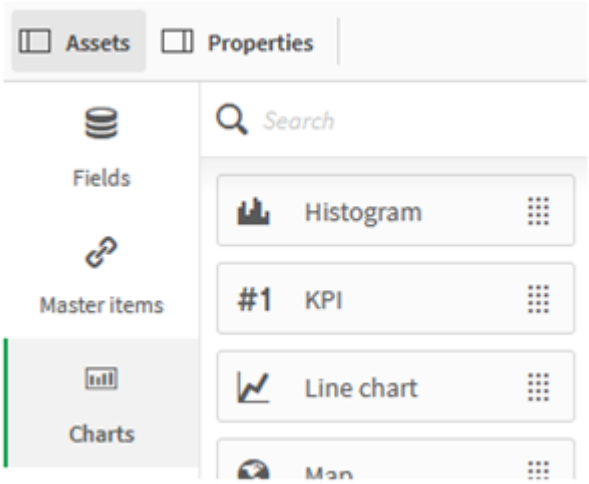
Der Modifikator ist:

```
<Year = {2015}>
```

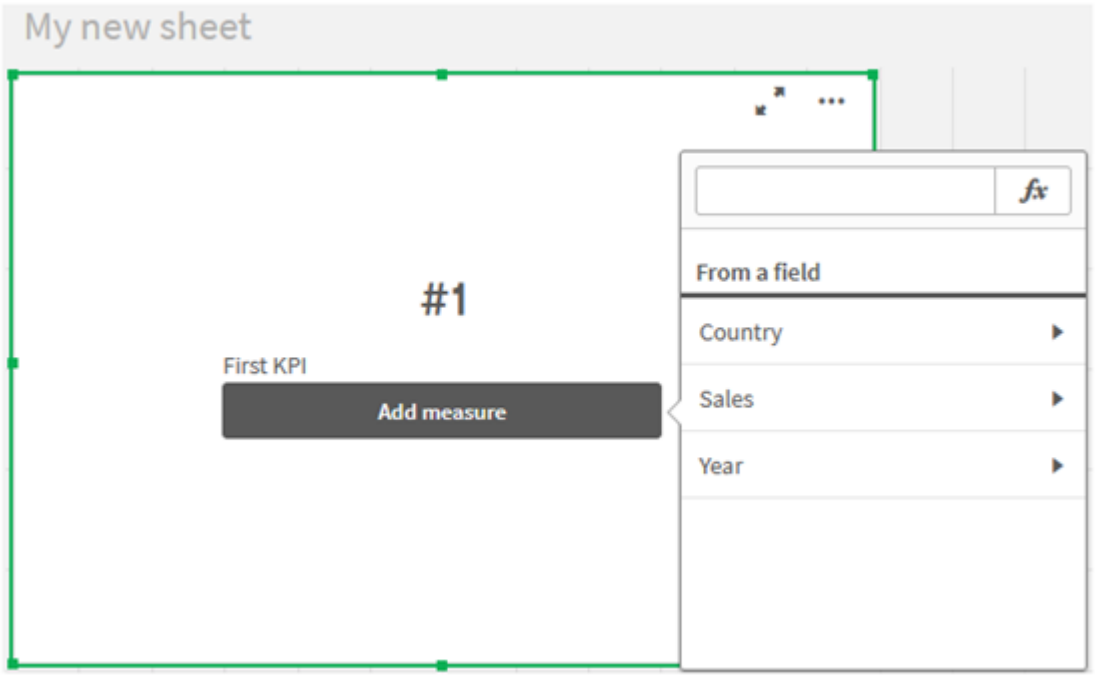
Dieser Modifikator gibt an, dass Daten aus dem Jahr 2015 ausgewählt werden. Die geschweiften Klammern, in die der Modifikator eingeschlossen ist, geben eine Auswahlformel an.

Gehen Sie folgendermaßen vor:

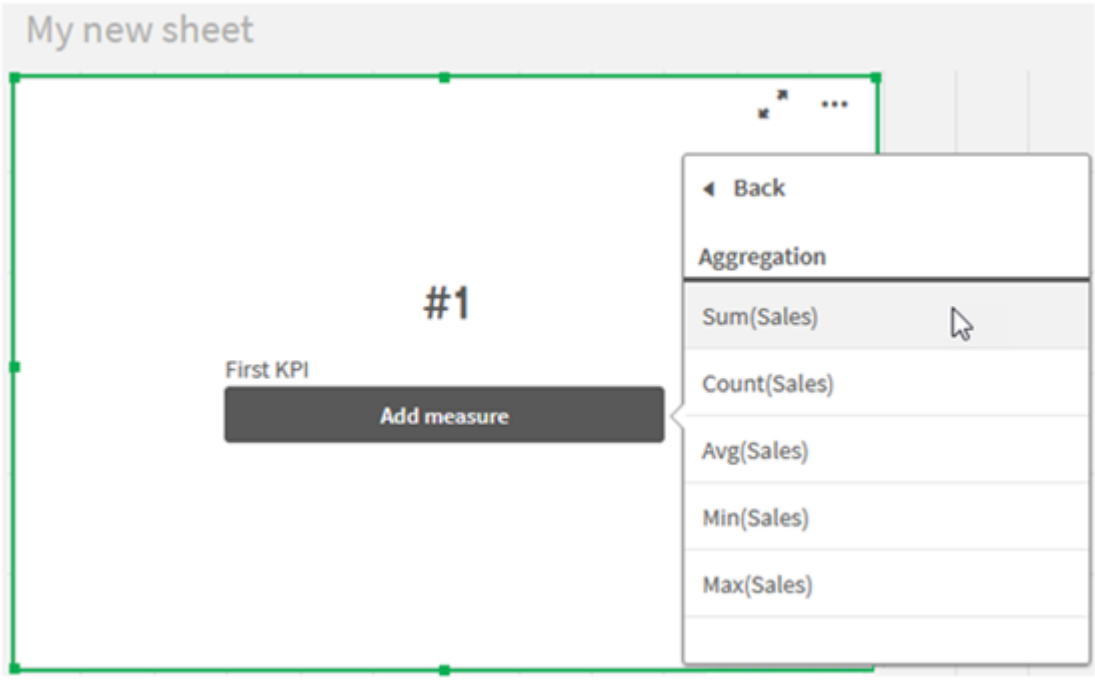
1. Öffnen Sie in einem Arbeitsblatt über die Navigationsleiste das **Extras**-Fenster und klicken Sie dann auf **Diagramme**.



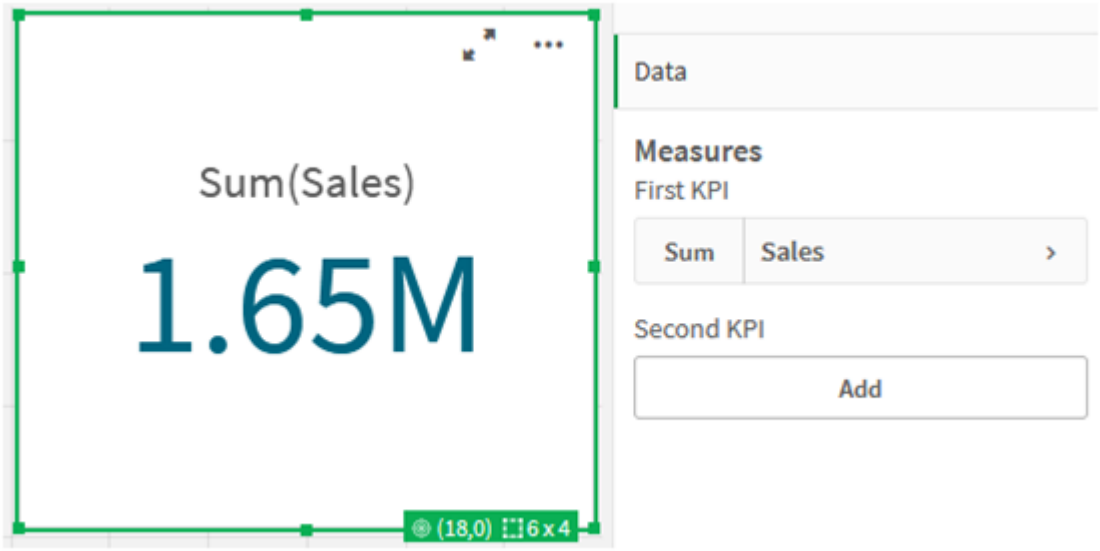
2. Ziehen Sie einen **KPI** auf das Arbeitsblatt und klicken Sie dann auf **Kennzahl hinzufügen**.



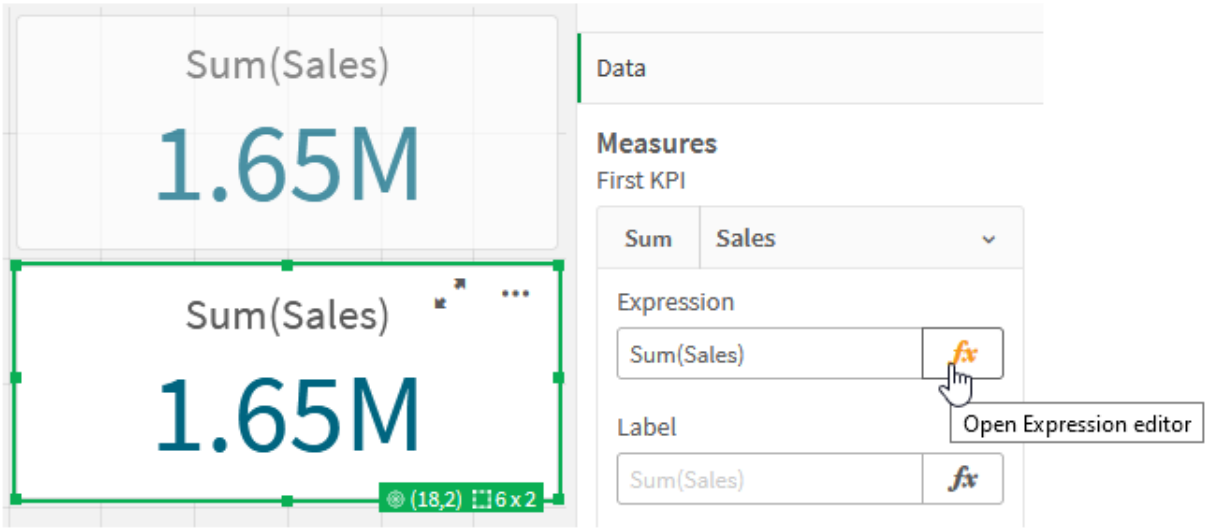
3. Klicken Sie auf `Sales` und wählen Sie `Sum(Sales)` für die Aggregierung aus.



Der KPI zeigt die Summe der Umsätze für alle Jahre.

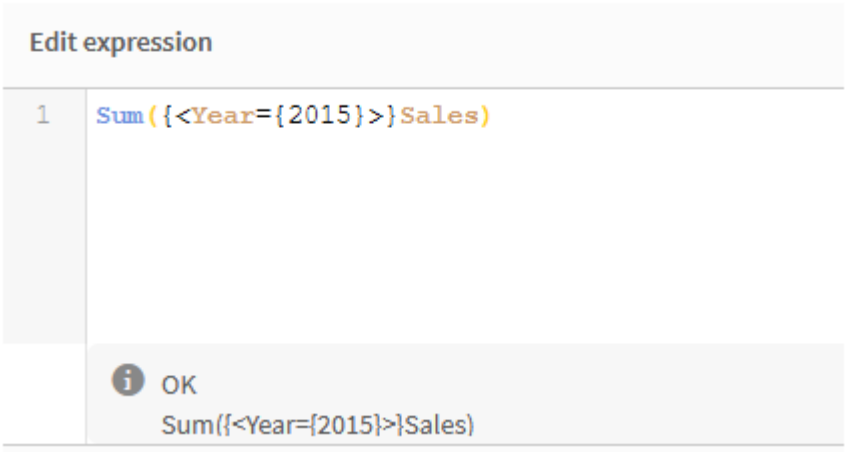


4. Kopieren Sie den KPI und fügen Sie ihn ein, um einen neuen KPI zu erstellen.
5. Klicken Sie auf den neuen KPI, klicken Sie unter **Kennzahlen** auf **Sales** und klicken Sie dann auf **Formel-Editor öffnen**.

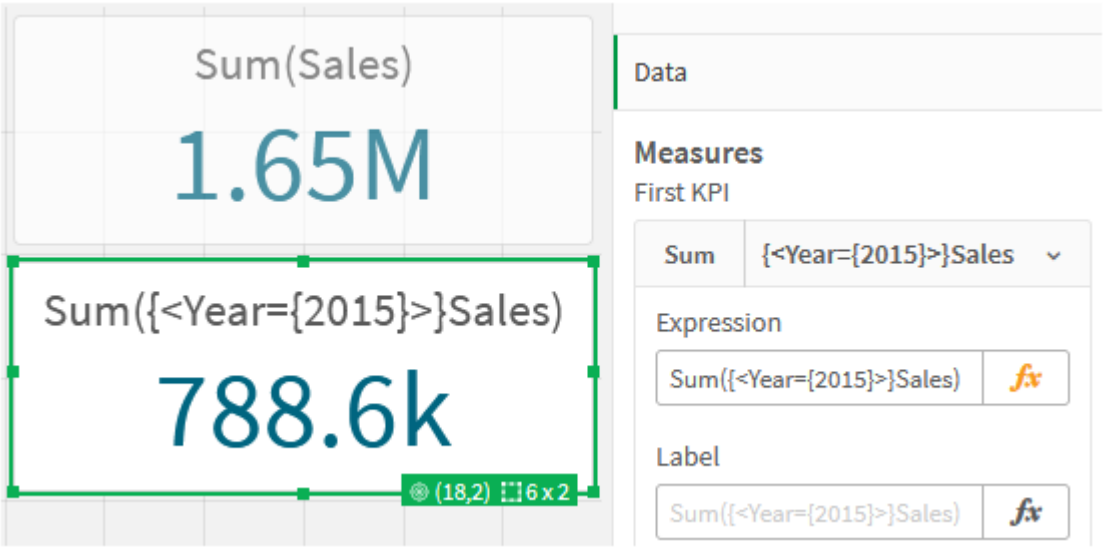


Der Formel-Editor wird mit der Aggregierung `Sum(Sales)` geöffnet.

6. Erstellen Sie im Formel-Editor eine Formel, um nur Sales für 2015 zu summieren:
   i. Fügen Sie geschweifte Klammern hinzu, um eine Auswahlformel anzugeben: `Sum({}Sales)`
   i. Fügen Sie spitze Klammern hinzu, um einen Modifikator für Auswahlformeln anzugeben: `Sum ({<>}Sales)`
   ii. Fügen Sie in den spitzen Klammern das auszuwählende Feld hinzu, in diesem Fall `Year`, gefolgt von einem Gleichheitszeichen. Schließen Sie dann 2015 in einen weiteren Satz geschweifte Klammern ein. Daraus ergibt sich der Modifikator für Auswahlformeln: `{<Year={2015}>}`.
   Die ganze Formel lautet:
   `Sum({<Year={2015}>}Sales)`

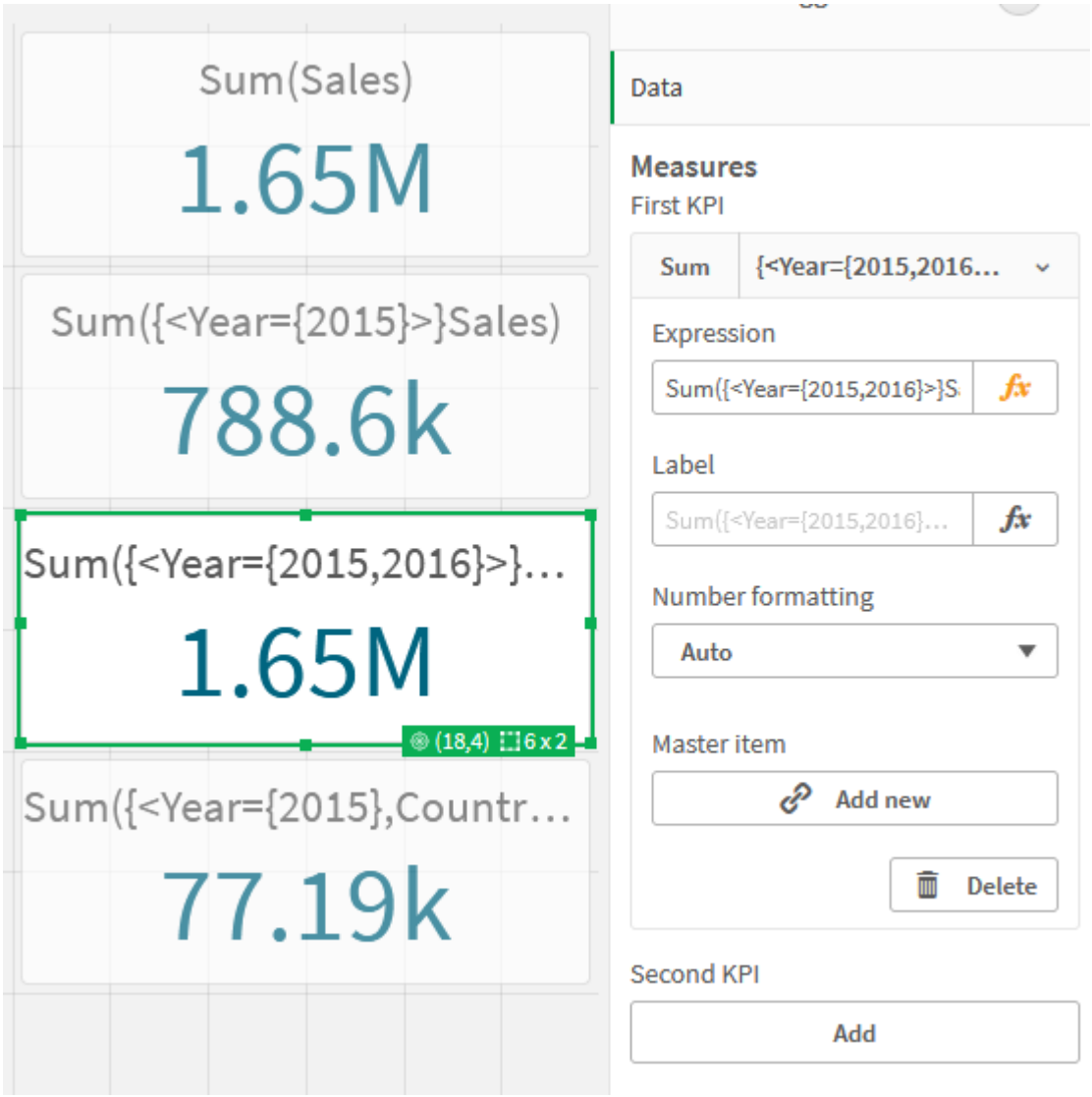


   iii. Klicken Sie auf **Anwenden**, um die Formel zu speichern und den Formel-Editor zu schließen. Die Summe von Sales für 2015 wird im KPI gezeigt.



7. Erstellen Sie zwei weitere KPIs mit den folgenden Formeln:
   `Sum({<Year={2015,2016}>}Sales)`
   Der Modifikator in der obigen Formel ist `<Year={2015,2016}>`. Die Formel gibt die Summe für Sales für 2015 und 2016 zurück.
   `Sum({<Year={2015},Country={'Germany'}>} Sales)`
   Der Modifikator in der obigen Formel ist `<Year={2015}, Country={'Germany'}>`. Die Formel gibt die Summe für Sales für 2015 zurück, wobei sich 2015 mit Germany schneidet.

KPIs mit Modifikatoren für Auswahlformeln



Hinzufügen von Identifikatoren für Auswahlformeln

In den obigen Auswahlformeln werden die aktuellen Auswahlen als Basis verwendet, weil kein Identifikator verwendet wurde. Fügen Sie dann Identifikatoren hinzu, um das Verhalten bei Treffen von Auswahlen anzugeben.

Gehen Sie folgendermaßen vor:

Erstellen oder kopieren Sie in Ihrem Arbeitsblatt die folgenden Auswahlformeln:

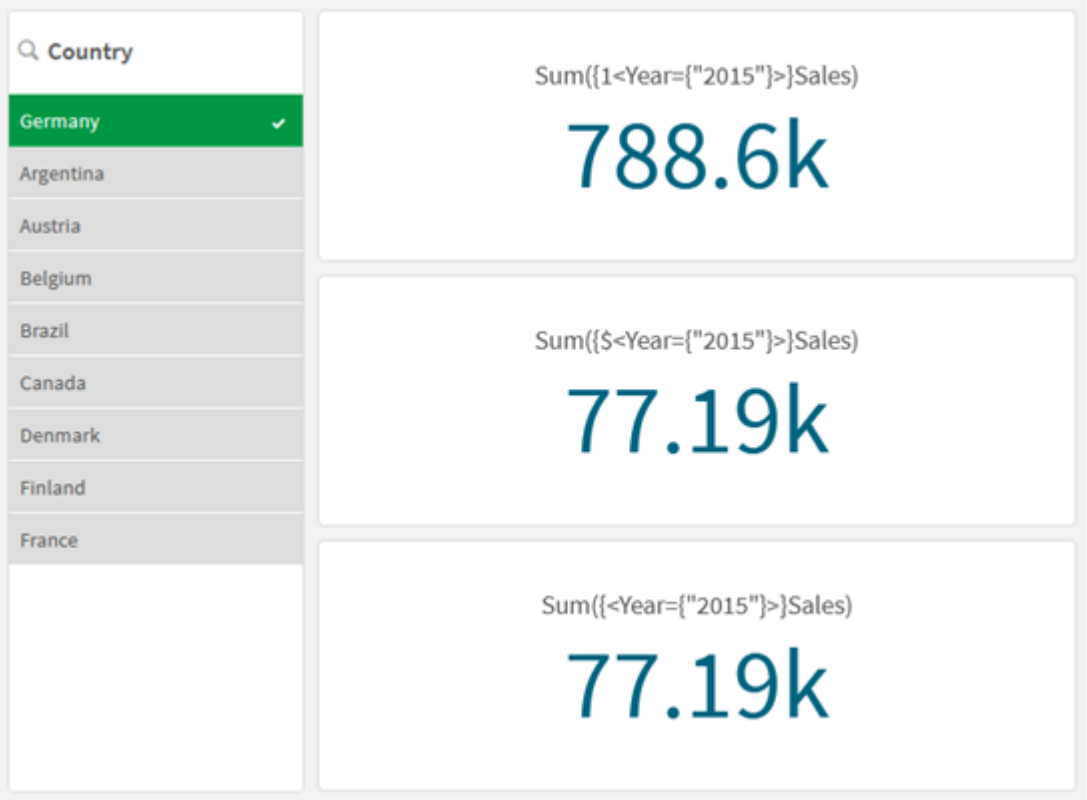
```
Sum({$<Year={"2015"}>}Sales)
```

Der Identifikator `$` basiert die Auswahlformel auf den aktuellen Auswahlen, die in den Daten getroffen wurden. Dies ist auch das Standardverhalten, wenn kein Identifikator verwendet wird.

```
Sum({1<Year={"2015"}>}Sales)
```

Der Identifikator 1 sorgt dafür, dass die Aggregierung von `Sum(Sales)` für 2015 die aktuelle Auswahl ignoriert. Der Wert der Aggregierung ändert sich nicht, wenn der Benutzer andere Auswahlen trifft. Wenn beispielsweise unten Germany ausgewählt wird, ändert sich der Wert der aggregierten Summe von 2015 nicht.

KPIs mit Modifikatoren und Identifikatoren für Auswahlformeln



Hinzufügen von Operatoren

Operatoren für Auswahlformeln werden zum Einschließen, Ausschließen und Überschneiden von Datensätzen verwendet. Sie benutzen Auswahlformeln als Operanden und liefern wiederum eine Auswahlformel.

Sie können Operatoren für Auswahlformeln in verschiedenen Situationen einsetzen:

- Zum Durchführen eines Vorgangs für Identifikatoren für Auswahlformeln, der Sätze von Datensätzen in Daten darstellt.
- Zum Durchführen eines Vorgangs für die Elementsätze, die Feldwerte oder innerhalb eines Modifikators für Auswahlformeln.

Gehen Sie folgendermaßen vor:

Erstellen oder kopieren Sie in Ihrem Arbeitsblatt die folgende Auswahlformel:

```
Sum({$<Year={2015}>+1<Country={'Germany'}>}Sales)
```

Hier sorgt das Pluszeichen (`(+)`) für eine Verbindung der Datensätze für 2015 und Germany. Wie für Identifikatoren für Auswahlformeln oben erläutert, bedeutet der Dollarzeichen-Identifikator (`($)`), dass die aktuellen Auswahlen für den ersten Operanden (`<Year={2015}>`) respektiert werden. Der Identifikator `1` bedeutet, dass die Auswahl für den zweiten Operanden (`<Country={'Germany'}>`) ignoriert wird.

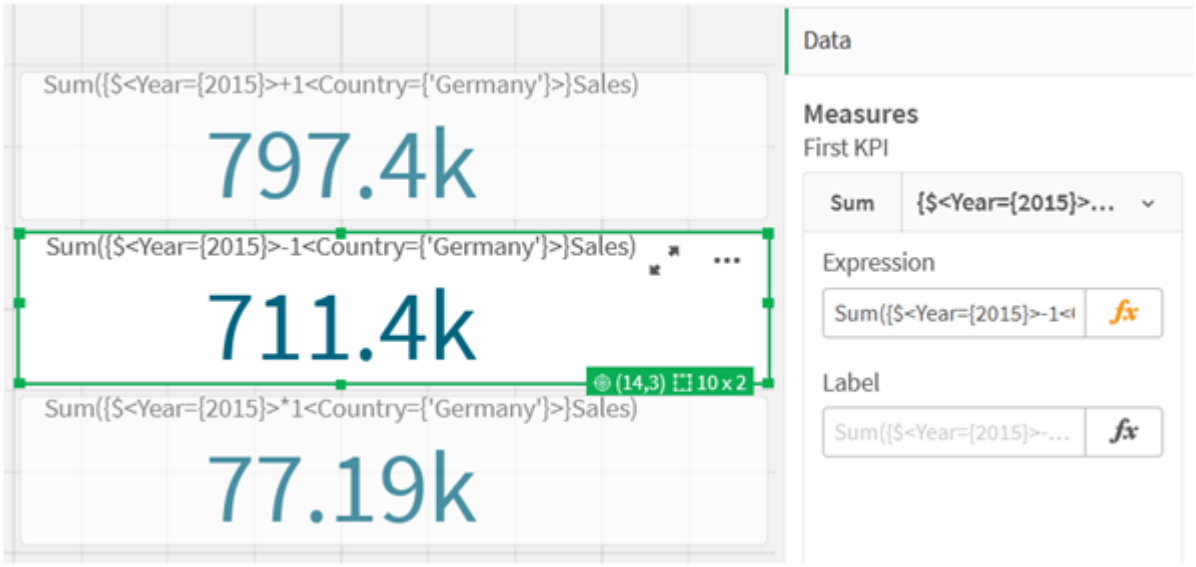
KPI mit dem Pluszeichen-Operator (+)

Alternativ können Sie ein Minuszeichen (`(-)`) verwenden, um einen Datenbestand zurückzugeben, der aus den Datensätzen besteht, die zu 2015, aber nicht zu Germany gehören. Oder verwenden Sie einen Asterisk (`(*)`), um einen Satz aus den Datensätzen zurückzugeben, die zu beiden Sätzen gehören.

```
Sum({$<Year={2015}>-1<Country={'Germany'}>}Sales)
```

```
Sum({$<Year={2015}>*1<Country={'Germany'}>}Sales)
```

KPIs mit Operatoren



Tutorial-Daten für Auswahlformeln

Ladeskript

Laden Sie die folgenden Daten als Inline-Ladevorgang und erstellen Sie dann die Diagrammformeln im Tutorial.

```
//Create table SalesByCountry
SalesByCountry:
Load * Inline [
Country, Year, Sales
Argentina, 2016, 66295.03
Argentina, 2015, 140037.89
Austria, 2016, 54166.09
Austria, 2015, 182739.87
Belgium, 2016, 182766.87
Belgium, 2015, 178042.33
Brazil, 2016, 174492.67
Brazil, 2015, 2104.22
Canada, 2016, 101801.33
Canada, 2015, 40288.25
Denmark, 2016, 45273.25
Denmark, 2015, 106938.41
Finland, 2016, 107565.55
Finland, 2015, 30583.44
France, 2016, 115644.26
France, 2015, 30696.98
Germany, 2016, 8775.18
Germany, 2015, 77185.68
];
```

Syntax für Auswahlformeln

Die allgemeine Syntax (optionale Klammern für eine mögliche Gliederung des Ausdrucks sind nicht berücksichtigt) wird mit dem Backus-Naur-Formalismus beschrieben:

```
set_expression ::= { set_entity { set_operator set_entity } }
set_entity ::= set_identifier [ set_modifier ] | set_modifier
set_identifier ::= 1 | $ | $N | $_N | bookmark_id | bookmark_name
set_operator ::= + | - | * | /
set_modifier ::= < field_selection {, field_selection } >
field_selection ::= field_name [ = | += | -= | *= | /= ] element_set_
expression
element_set_expression ::= [ - ] element_set { set_operator element_set }
element_set ::= [ field_name ] | { element_list } | element_function
element_list ::= element { , element }
element_function ::= ( P | E ) ( [set_expression] [field_name] )
element ::= field_value | " search_mask "
```

6.3 Allgemeine Syntax für Diagrammformeln

Die folgende allgemeine Syntaxstruktur kann zusammen mit vielen optionalen Parametern für Diagrammformeln verwendet werden:

```
expression ::= ( constant | expressionname | operator1 expression | expression operator2
expression | function | aggregation function | (expression ) )
```

Dabei gilt:

constant ist ein String (Text, Datum oder Uhrzeit) in einfachen geraden Anführungszeichen oder eine Zahl. Konstanten werden ohne Tausendertrennzeichen und mit einem Punkt als Dezimaltrennzeichen geschrieben.

expressionname ist die Bezeichnung einer anderen Formel innerhalb desselben Diagramms.

operator1 ist ein einwertiger Operator. Er bezieht sich auf eine einzige Formel, die rechts vom Operator steht.

operator2 ist ein zweiwertiger Operator. Er bezieht sich auf zwei Formeln, eine links und eine rechts vom Operator.

```
function ::= functionname ( parameters )
parameters ::= expression { , expression }
```
Zahl und Art der Parameter sind nicht willkürlich, Sie hängen von der verwendeten Funktion ab.

```
aggregationfunction ::= aggregationfunctionname ( parameters2 )
parameters2 ::= aggrexpression { , aggrexpression }
```
Zahl und Art der Parameter sind nicht willkürlich, Sie hängen von der verwendeten Funktion ab.

6.4 Allgemeine Syntax für Aggregierungsformeln

Die folgende allgemeine Syntaxstruktur kann zusammen mit vielen optionalen Parametern für Aggregierungen verwendet werden:

```
aggrexpression ::= ( fieldref | operator1 aggrexpression | aggrexpression operator2
aggrexpression | functioninaggr | ( aggrexpression ) )
```
fieldref ist ein Feldname.

```
functionaggr ::= functionname ( parameters2 )
```
Formeln und Funktionen können beliebig angeordnet werden, sofern **fieldref** immer in genau einer Aggregierungsfunktion eingeschlossen ist und die Formel sinnvoll interpretierbare Ergebnisse liefert. Nur bei fehlerhafter Syntax gibt Qlik Sense Fehlermeldungen aus.

7 Operatoren

In diesem Abschnitt werden die Operatoren beschreiben, die in Qlik Sense verwendet werden können. Es gibt zwei verschiedene Arten von Operatoren:

- Einwertige Operatoren (benötigen nur einen Operanden)
- Zweiwertige Operatoren (benötigen zwei Operanden)

Die meisten Operatoren sind zweiwertig.

Die Operatoren lassen sich in folgende Gruppen einteilen:

- Bit-Operatoren
- Logische Operatoren
- Numerische Operatoren
- Relationale Operatoren
- String-Operatoren

7.1 Bit-Operatoren

Die Bit-Operatoren konvertieren die Operanden in ganze Zahlen in der 32-Bit-Darstellung (Verkürzung) und liefern auch ebensolche Ergebnisse. Die Operationen werden Bit für Bit durchgeführt. Kann ein Operand nicht als Zahl interpretiert werden, liefert der Vorgang NULL.

Bit-Operatoren

Operator	Vollständiger Name	Beschreibung
bitnot	Inverses auf Bit-Ebene.	Einwertiger Operator. Die Operation liefert Bit für Bit das logische Gegenteil des Operanden. **Beispiel:** `bitnot 17` liefert -18
bitand	Logisches Und auf Bit-Ebene.	Die Operation liefert Bit für Bit das logische Und der Operanden. **Beispiel:** `17 bitand 7` liefert 1
bitor	Logisches Oder auf Bit-Ebene.	Die Operation liefert Bit für Bit das logische Oder der Operanden. **Beispiel:** `17 bitor 7` liefert 23

Operator	Vollständiger Name	Beschreibung
bitxor	Ausschließendes logisches Oder auf Bit-Ebene.	Die Operation liefert Bit für Bit das ausschließende logische Oder der Operanden. **Beispiel:** `17 bitxor 7` liefert 22
>>	Bit-weises Verschieben nach rechts.	Die Operation liefert den ersten um ein Bit nach rechts verschobenen Operanden. Die Anzahl Schritte ist im zweiten Operanden definiert. **Beispiel:** `8 >> 2` liefert 2
<<	Bit-weises Verschieben nach links.	Die Operation liefert den ersten um ein Bit nach linkt verschobenen Operanden. Die Anzahl Schritte ist im zweiten Operanden definiert. **Beispiel:** `8 << 2` liefert 32

7.2 Logische Operatoren

Logische Operatoren interpretieren die Operanden logisch und liefern als Ergebnis True (-1) oder False (0).

Logische Operatoren

Operator	Beschreibung
not	Logisches Gegenteil. Einwertiger Operator. Die Operation liefert das logische Gegenteil des Operanden.
and	Logisches Und. Die Operation liefert das logische Und der beiden Operanden.
or	Logisches Oder. Die Operation liefert das logische Oder der beiden Operanden.
Xor	Ausschließendes logisches Oder. Die Operation liefert das ausschließende logische Oder der beiden Operanden. Dies entspricht dem normalen logischen Oder mit der Ausnahme, dass das Ergebnis False lautet, wenn beide Operanden True sind.

7.3 Numerische Operatoren

Alle numerischen Operatoren benutzen die numerischen Werte der Operanden und liefern als Ergebnis einen numerischen Wert.

Numerische Operatoren

Operator	Beschreibung
+	Zeichen für eine positive Zahl (als einwertiger Operator) oder für die arithmetische Addition. Der zweiwertige Operator liefert die Summe der beiden Operanden.
-	Zeichen für eine negative Zahl (als einwertiger Operator) oder für die arithmetische Subtraktion. Der einwertige Operator liefert das Produkt des Operanden mit -1. Der zweiwertige Operator liefert die Differenz der beiden Operanden.
*	Arithmetische Multiplikation. Die Operation liefert das Produkt der beiden Operanden.
/	Arithmetische Division. Die Operation liefert den Quotienten bzw. das Verhältnis der beiden Operanden.

7.4 Relationale Operatoren

Relationale Operatoren vergleichen die beiden Operanden und liefern als Ergebnis True (-1) oder False (0). Alle relationalen Operatoren sind zweiwertig.

Relationale Operatoren

Operator	Beschreibung
<	Weniger als. Sind beide Operanden numerisch interpretierbar, findet ein numerischer Vergleich statt. Die Operation liefert das logische Ergebnis des Wertevergleichs.
<=	Kleiner oder gleich. Sind beide Operanden numerisch interpretierbar, findet ein numerischer Vergleich statt. Die Operation liefert das logische Ergebnis des Wertevergleichs.
>	Größer als. Sind beide Operanden numerisch interpretierbar, findet ein numerischer Vergleich statt. Die Operation liefert das logische Ergebnis des Wertevergleichs.
>=	Größer oder gleich. Sind beide Operanden numerisch interpretierbar, findet ein numerischer Vergleich statt. Die Operation liefert das logische Ergebnis des Wertevergleichs.
=	Gleich. Sind beide Operanden numerisch interpretierbar, findet ein numerischer Vergleich statt. Die Operation liefert das logische Ergebnis des Wertevergleichs.
<>	Ungleich. Sind beide Operanden numerisch interpretierbar, findet ein numerischer Vergleich statt. Die Operation liefert das logische Ergebnis des Wertevergleichs.

Operator	Beschreibung
precedes	Im Gegensatz zum Operator **<** wird vor dem Vergleich keine numerische Interpretation versucht. Liefert wahr, wenn der Operand auf der linken Seite eine Textdarstellung hat, die im Stringvergleich vor der Textdarstellung des rechten Operanden kommt. **Beispiel:** `'1 ' precedes ' 2'` liefert FALSE `' 1' precedes ' 2'` liefert TRUE da der ASCII-Wert eines Leerzeichens (' ') geringer ist als der ASCII-Wert einer Zahl. Aber: `'1 ' < ' 2'` liefert TRUE `' 1' < ' 2'` liefert TRUE
follows	Im Gegensatz zum Operator **>** wird vor dem Vergleich keine numerische Interpretation der Argumentwerte versucht. Liefert wahr, wenn der Operand auf der linken Seite eine Textdarstellung hat, die im Stringvergleich nach der Textdarstellung des rechten Operanden kommt. **Beispiel:** `' 2' follows '1'` liefert FALSE ' 2' follows ' 1' liefert TRUE da der ASCII-Wert eines Leerzeichens (' ') geringer ist als der ASCII-Wert einer Zahl. Aber: `' 2' > ' 1'` liefert TRUE `' 2' > '1 '` liefert TRUE

7.5 String-Operatoren

Es existieren zwei String-Operatoren. Der erste liefert den aus beiden Operanden zusammengesetzten String, der zweite führt einen Vergleich beider Operanden durch und liefert als Ergebnis einen booleschen Wert.

&

String-Verkettung. Die Operation liefert einen String, der aus den beiden Operanden-Strings in der gegebenen Reihenfolge zusammengesetzt ist.

Beispiel:

`'abc' & 'xyz'` liefert 'abcxyz'

like

String-Vergleich mit Wildcards. Die Operation liefert ein boolesches True (-1), wenn der erste Operator dem Suchmuster des zweiten entspricht. Der zweite String darf die Wildcards * (für beliebig viele beliebige Zeichen) oder ? (ein beliebiges Zeichen) enthalten.

Beispiel:

`'abc' like 'a*'` liefert True (-1)

`'abcd' like 'a?c*'` liefert True (-1)

`'abc' like 'a??bc'` liefert False (0)

8 Skript- und Diagrammfunktionen

Wandeln Sie Daten anhand von Funktionen in Datenladeskripts und Diagrammformeln um und aggregieren Sie die Daten.

Viele Funktionen können sowohl in Datenladeskripts und Diagrammformeln auf die gleiche Weise verwendet werden, aber es gibt eine Reihe von Ausnahmen:

- Einige Funktionen können nur in Datenladeskripts verwendet werden, gekennzeichnet durch Skriptfunktion.
- Einige Funktionen können nur in Diagrammformeln verwendet werden, gekennzeichnet durch Diagrammfunktion.
- Einige Funktionen können sowohl in Datenladeskripten als auch in Diagrammformeln verwendet werden, aber mit Unterschieden bei Parametern und in der Anwendung. Diese werden als separate Themen, gekennzeichnet durch Skriptfunktion oder Diagrammfunktion, beschrieben.

8.1 Analyseverbindungen für serverseitige Erweiterungen (SSE)

Durch Analyseverbindungen aktivierte Funktionen sind nur dann sichtbar, wenn Sie die Analyseverbindungen konfiguriert haben und Qlik Sense gestartet wurde.

Analyseverbindungen werden in der QMC konfiguriert. Weitere Informationen finden Sie im Abschnitt „Erstellen von Analyseverbindungen“ im Handbuch Qlik Sense Sites verwalten.

In Qlik Sense Desktop konfigurieren Sie die Analyseverbindungen durch Bearbeiten der Datei *Settings.ini*. Weitere Informationen finden Sie im Abschnitt „Konfigurieren von Analyseverbindungen in Qlik Sense Desktop“ im Handbuch Qlik Sense Desktop.

8.2 Aggregierungsfunktionen

Die als Aggregierungsfunktionen bezeichnete Funktionsgruppe besteht aus Funktionen, die mehrere Feldwerte als Input verwenden und ein einzelnes Ergebnis pro Gruppe ausgeben, wobei die Gruppierung von einer Diagrammdimension oder von einer **group by**-Bedingung in der Skriptanweisung definiert wird.

Aggregierungsfunktionen umfassen **Sum()**, **Count()**, **Min()**, **Max()** und viele weitere.

Die meisten Aggregierungsfunktionen können sowohl im Datenladeskript als auch in den Diagrammformeln verwendet werden, jedoch variiert die Syntax.

Beschränkungen:

Die Parameter der Aggregierungsfunktion dürfen keine anderen Aggregierungsfunktionen enthalten, es sei denn, diese inneren Aggregierungen enthalten den Qualifizierer **TOTAL**. Für komplexere verschachtelte Aggregierungen verwenden Sie die erweiterte Funktion **Aggr** in Verbindung mit einer angegebenen Dimension.

Wenn Sie ein Element benennen, müssen Sie vermeiden, den gleichen Namen mehr als einem Feld, einer Variablen oder einer Kennzahl zuzuweisen. Beim Auflösen von Konflikten zwischen Elementen mit identischen Namen wird eine strikte Reihenfolge eingehalten. Diese Reihenfolge gilt auch bei allen Objekten oder Kontexten, in denen diese Elemente verwendet werden. Diese Reihenfolge lautet wie folgt:

- Innerhalb einer Aggregierung hat ein Feld Vorrang vor einer Variablen. Kennzahlbezeichnungen sind in Aggregierungen nicht relevant und haben keine Priorität.
- Außerhalb einer Aggregierung hat eine Kennzahlbezeichnung Vorrang vor einer Variablen, die wiederum Vorrang vor einem Feldnamen hat.
- Zudem kann außerhalb einer Aggregierung eine Kennzahl wiederverwendet werden, indem ihre Bezeichnung referenziert wird, es sei denn, es handelt sich um eine berechnete Bezeichnung. In dieser Situation reduziert sich die Bedeutung der Kennzahl, um das Risiko eines Selbstbezugs zu verringern, und in diesem Fall wird der Name immer zuerst als Kennzahlbezeichnung interpretiert, an zweiter Stelle als Feldname und an dritter Stelle als Variablenname.

Aggregierungsfunktionen im Datenladeskript verwenden

Aggregierungsfunktionen können nur innerhalb von **LOAD** - und **SELECT**-Befehlen verwendet werden.

Aggregierungsfunktionen in Diagrammformeln verwenden

Die Parameter der Aggregierungsfunktion dürfen keine anderen Aggregierungsfunktionen enthalten, es sei denn, diese inneren Aggregierungen enthalten den Qualifizierer **TOTAL**. Für komplexere verschachtelte Aggregierungen verwenden Sie die erweiterte Funktion **Aggr** in Verbindung mit einer angegebenen Dimension.

Die Aggregierungsfunktion berechnet sich über alle wählbaren Werte. Alternativ können Sie die der Berechnung zugrunde liegenden Werte auch über eine Auswahlformel bestimmen.

Berechnung von Aggregierungen

Eine Aggregierung geht in einer Schleife die Datensätze einer bestimmten Tabelle durch und aggregiert die enthaltenen Datensätze. Beispielsweise zählt **Count**(<Field>) die Anzahl der Datensätze in der Tabelle, in der sich <Field> befindet. Wenn Sie nur die distinkten Feldwerte aggregieren möchten, müssen Sie den **distinct**-Befehl verwenden, beispielsweise **Count**(**distinct** <Field>).

Wenn die Aggregierungsfunktion Felder aus verschiedenen Tabellen enthält, geht die Aggregierungsfunktion in einer Schleife die Datensätze des Kreuzprodukts der Tabellen der sie bildenden Felder durch. Dies beeinträchtigt die Leistung, und aus diesem Grund sollten derartige Aggregierungen vermieden werden, besonders wenn es sich um große Datenmengen handelt.

Aggregierung von Schlüsselfeldern

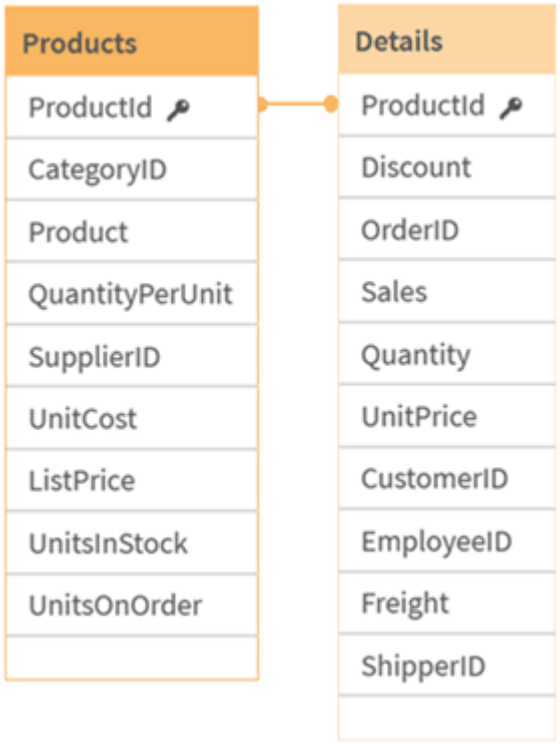
Die Art und Weise, wie Aggregierungen berechnet werden, bedeutet, dass keine Schlüsselfelder aggregiert werden können, da nicht klar ist, welche Tabelle für die Aggregierung verwendet werden sollte. Wenn zum Beispiel das Feld <Key> zwei Tabellen verknüpft, ist nicht klar, ob **Count**(<Key>) die Anzahl der Datensätze aus der ersten oder aus der zweiten Tabelle zurückgeben sollte.

Wenn Sie dagegen den **distinct**-Befehl verwenden, ist die Aggregierung korrekt definiert und kann berechnet werden.

Wenn Sie also ein Schlüsselfeld in einer Aggregierungsfunktion ohne den **distinct**-Befehl verwenden, gibt Qlik Sense eine Zahl zurück, die möglicherweise keinen Sinn ergibt. Die Lösung besteht darin, entweder den **distinct**-Befehl zu verwenden oder eine Kopie des Schlüssels zu verwenden, die sich nur in einer Tabelle befindet.

Beispielsweise ist in den folgenden Tabellen ProductID der Schlüssel zwischen den Tabellen.

ProductID-Schlüssel zwischen den Tabellen „Products“ und „Details“



Count(ProductID) kann entweder in der Tabelle Products (mit einem Datensatz pro Produkt – ProductID ist der primäre Schlüssel) oder in der Tabelle Details (die wahrscheinlich mehrere Datensätze pro Produkt enthält) gezählt werden. Um die Anzahl der unterschiedlichen Produkte zu ermitteln, sollten Sie Count(distinct ProductID) verwenden. Wenn Sie die Anzahl der Zeilen in einer bestimmten Tabelle zählen möchten, verwenden Sie den Schlüssel nicht.

Einfache Aggregierungsfunktionen

Einfache Aggregierungsfunktionen – Übersicht

Die einfachen Aggregierungsfunktionen bilden die Gruppe der am häufigsten anzutreffenden Aggregierungsfunktionen.

Jede Funktion wird nach der Übersicht genauer beschrieben. Sie können auch auf den Funktionsnamen in der Syntax klicken, um direkt auf die Details zu der spezifischen Funktion zuzugreifen.

Einfache Aggregierungsfunktionen im Datenladeskript verwenden

FirstSortedValue

FirstSortedValue() gibt den Wert der in **value** festgelegten Formel zurück, der dem Ergebnis der Sortierung des **sort_weight**-Arguments entspricht, z. B. der Name des Produkts mit dem niedrigsten Preis pro Einheit. Der n-te-Wert in der Sortierreihenfolge kann in **rank** festgelegt werden. Weist mehr als ein Ergebnis dasselbe Feld **sort_weight** für die festgelegte Funktion **rank** auf, gibt die Funktion NULL zurück. Die sortierten Werte werden über eine Reihe von Datensätzen iteriert, wie von einer **group by**-Bedingung definiert, oder über einen kompletten Datensatz aggregiert, wenn keine **group by**-Bedingung definiert ist.

```
FirstSortedValue ([ distinct ] expression, sort_weight [, rank ])
```

Max

Max() liefert den höchsten numerischen Wert der aggregierten Daten in der Formel, wie in einer Bedingung **group by** definiert wurde. Durch die Festlegung eines **rank** n kann der höchste n-te Wert gefunden werden.

```
Max ( expression[, rank])
```

Min

Min() liefert den niedrigsten numerischen Wert der aggregierten Daten in der Formel, wie in einer Bedingung **group by** definiert wurde. Durch die Festlegung eines **rank** n kann der niedrigste n-te Wert gefunden werden.

```
Min ( expression[, rank])
```

Mode

Mode() liefert den am häufigsten vorkommenden Wert, also den Moduswert der aggregierten Daten in der Formel, wie in einer Bedingung **group by** definiert wurde. Die Funktion **Mode()** kann sowohl numerische Werte als auch Textwerte liefern.

```
Mode (expression )
```

Only

Only() liefert einen Wert, wenn die aggregierten Daten nur ein einziges mögliches Ergebnis liefern. Enthält ein Datensatz nur einen einzigen Wert, wird dieser Wert geliefert, ansonsten NULL. Verwenden Sie die Bedingung **group by** für die Evaluierung über mehrere Datensätze. Die Funktion **Only()** kann numerische und Textwerte liefern.

```
Only (expression )
```

Sum

Sum() berechnet den Gesamtwert der in der Formel aggregierten Werte, wie in einer Bedingung **group by** definiert wurde.

```
Sum ([distinct]expression)
```

Einfache Aggregierungsfunktionen in Diagrammformeln verwenden

Diagramm-Aggregierungsfunktionen können ausschließlich in den Formeln von Diagrammen verwendet werden. Aggregierungsfunktionen dürfen nicht verschachtelt werden, d. h. das Argument einer Aggregierungsfunktion darf keine weiteren Aggregierungsfunktionen enthalten.

FirstSortedValue

FirstSortedValue() gibt den Wert der in **value** festgelegten Formel zurück, der dem Ergebnis der Sortierung des **sort_weight**-Arguments entspricht, z. B. der Name des Produkts mit dem niedrigsten Preis pro Einheit. Der n-te-Wert in der Sortierreihenfolge kann in **rank** festgelegt werden. Weist mehr als ein Ergebnis dasselbe Feld **sort_weight** für die festgelegte Funktion **rank** auf, gibt die Funktion NULL zurück.

```
FirstSortedValue - Diagrammfunktion([{SetExpression}] [DISTINCT] [TOTAL [<fld
{,fld}>]] value, sort_weight [,rank])
```

Max

Max() liefert den höchsten Wert der aggregierten Daten. Durch die Festlegung eines **rank** n kann der höchste n-te Wert gefunden werden.

```
Max - DiagrammfunktionMax() liefert den höchsten Wert der aggregierten Daten.
Durch die Festlegung eines rank n kann der höchste n-te Wert gefunden werden.
Es ist auch ratsam, sich FirstSortedValue und rangemax anzusehen. Diese haben
eine ähnliche Funktion wie die Funktion Max. Max([{SetExpression}] [TOTAL
[<fld {,fld}>]] expr [,rank]) numerisch ArgumenteArgumentBeschreibungexprDie
Formel oder das Feld mit den Daten, die gemessen werden sollen.rankDer
Standardwert von rank ist 1, was dem höchsten Wert entspricht. Wird rank als
2 angegeben, wird der zweithöchste Wert ausgegeben. Wird rank als 3
angegeben, wird der dritthöchste Wert ausgegeben
usw.SetExpressionStandardmäßig berechnet sich die Aggregierungsfunktion über
alle wählbaren Werte. Alternativ können Sie die der Berechnung zugrunde
liegenden Werte über die Auswahlformel bestimmen. TOTALDer Zusatz TOTAL vor
der Funktion bewirkt, dass die Berechnung über alle ausgewählten bzw.
wählbaren Werte erfolgt, und nicht nur über diejenigen, die zu dem Wert der
aktuellen Dimension zählen. Die Dimensionen des Diagramms werden also nicht
berücksichtigt. Mit TOTAL [<fld {.fld}>], wobei auf den Zusatz TOTAL eine
Liste aus mindestens einem Feldnamen (d. h. einer Teilmenge der
Diagrammdimensionsvariablen) folgt, erstellen Sie eine Teilmenge aller
möglichen Werte. DatenCustomerProductUnitSalesUnitPrice
AstridaAA416AstridaAA1015AstridaBB99BetacabBB510BetacabCC220BetacabDD-
25CanutilityAA815CanutilityCC-19Beispiele und
ErgebnisseBeispieleErgebnisseMax(UnitSales)10, da dies der höchste Wert in
UnitSales ist.Der Wert einer Bestellung wird aus der Anzahl der verkauften
Einheiten (UnitSales) multipliziert mit dem Stückpreis berechnet.Max
(UnitSales*UnitPrice)150, da dies der höchste Wert der Berechnung aus allen
wählbaren Werten von (UnitSales)*(UnitPrice) ist.Max(UnitSales, 2)9, da dies
der zweithöchste Wert ist.Max(TOTAL UnitSales)10, da der Zusatz TOTAL
bedeutet, dass der höchste wählbare Wert gefunden wird, ohne die Dimensionen
des Diagramms zu berücksichtigen. Bei einem Diagramm mit Customer als
Dimension sorgt der Zusatz TOTAL dafür, dass der maximale Wert übergreifend
über den vollständigen Datensatz anstelle des Maximalwerts UnitSales für
jeden Kunden zurückgegeben wird.Treffen Sie die Auswahl Customer B.Max({1}
TOTAL UnitSales)10, unabhängig von der getroffenen Auswahl, da die Set
Analysis-Formel {1} den Datensatz so definiert, dass er als ALL bewertet
wird, unabhängig davon, welche Auswahl getroffen wird.In Beispielen
```

```
verwendete Daten:ProductData:LOAD * inline
[Customer|Product|UnitSales|UnitPriceAstrida|AA|4|16Astrida|AA|10|15Astrida|B
B|9|9Betacab|BB|5|10Betacab|CC|2|20Betacab|DD||25Canutility|AA|8|15Canutility
|CC||19] (delimiter is '|'); FirstSortedValue RangeMax ([{SetExpression}]
[DISTINCT] [TOTAL [<fld {,fld}>]] expr [,rank])
```

Min

Min() liefert den niedrigsten Wert der aggregierten Daten. Durch die Festlegung eines **rank** n kann der niedrigste n-te Wert gefunden werden.

```
Min - Diagrammfunktion([{SetExpression}] [DISTINCT] [TOTAL [<fld {,fld}>]]
expr [,rank])
```

Mode

Mode() liefert den am häufigsten vorkommenden Wert, also den Moduswert, in den aggregierten Daten. Die Funktion **Mode()** kann sowohl Textwerte als auch numerische Werte verarbeiten.

```
Mode - Diagrammfunktion ({[SetExpression] [TOTAL [<fld {,fld}>]]} expr)
```

Only

Only() liefert einen Wert, wenn die aggregierten Daten nur ein einziges mögliches Ergebnis liefern. Wenn Sie beispielsweise nach dem einzigen Produkt suchen, bei dem der Preis pro Einheit 9 beträgt, wird NULL geliefert, wenn mehr als ein Produkt einen Preis pro Einheit von 9 aufweist.

```
Only - Diagrammfunktion([{SetExpression}] [DISTINCT] [TOTAL [<fld {,fld}>]]
expr)
```

Sum

Sum() berechnet die Gesamtsumme der von der Formel vorgegebenen Werte oder Felder über die aggregierten Daten.

```
Sum - Diagrammfunktion([{SetExpression}] [DISTINCT] [TOTAL [<fld {,fld}>]]
expr])
```

FirstSortedValue

FirstSortedValue() gibt den Wert der in **value** festgelegten Formel zurück, der dem Ergebnis der Sortierung des **sort_weight**-Arguments entspricht, z. B. der Name des Produkts mit dem niedrigsten Preis pro Einheit. Der n-te-Wert in der Sortierreihenfolge kann in **rank** festgelegt werden. Weist mehr als ein Ergebnis dasselbe Feld **sort_weight** für die festgelegte Funktion **rank** auf, gibt die Funktion NULL zurück. Die sortierten Werte werden über eine Reihe von Datensätzen iteriert, wie von einer **group by**-Bedingung definiert, oder über einen kompletten Datensatz aggregiert, wenn keine **group by**-Bedingung definiert ist.

Syntax:

```
FirstSortedValue ([ distinct ] value, sort-weight [, rank ])
```

Rückgabe Datentyp: dual

Argumente:

Argumente

Argument	Beschreibung
value Expression	Die Funktion findet den Wert der Formel **value**, der dem Ergebnis der Sortierung von **sort_weight** entspricht.
sort-weight Expression	Die Formel mit den Daten, die sortiert werden sollen. Der erste (niedrigste) Wert von **sort_weight** wird gefunden, aus dem der entsprechende Wert der Formel **value** bestimmt wird. Wenn Sie ein Minuszeichen vor **sort_weight** voranstellen, liefert die Funktion stattdessen den letzten (höchsten) sortierten Wert.
rank Expression	Durch **rank** "n" größer 1 wird der n-te Wert ausgegeben.
distinct	Der Zusatz **DISTINCT** vor den Funktionsargumenten bewirkt, dass bei der Auswertung der Funktionsargumente entstehende Duplikate nicht berücksichtigt werden.

Beispiele und Ergebnisse:

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Fügen Sie einem Arbeitsblatt in unserer App dann mindestens diejenigen Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um das Ergebnis anzuzeigen.

Damit die Anzeige genauso wie in der unteren Ergebnisspalte aussieht, schalten Sie im Eigenschaftsfenster unter "Sortierung" von "Auto" auf "Benutzerdefiniert" um und heben Sie anschließend die numerische und alphabetische Sortierung auf.

Skriptbeispiele

Beispiel	Ergebnis
Temp: LOAD * inline [Customer|Product|OrderNumber|UnitSales|CustomerID Astrida|AA|1|10|1 Astrida|AA|7|18|1 Astrida|BB|4|9|1 Astrida|CC|6|2|1 Betacab|AA|5|4|2 Betacab|BB|2|5|2 Betacab|DD|12|25|2 Canutility|AA|3|8|3 Canutility|CC|13|19|3 Divadip|AA|9|16|4 Divadip|AA|10|16|4 Divadip|DD|11|10|4] (delimiter is '|'); FirstSortedValue: LOAD Customer,FirstSortedValue(Product, UnitSales) as MyProductWithSmallestOrderByCustomer Resident Temp Group By Customer;	Customer MyProductWithSmallestOrderByCustomer Astrida CC Betacab AA Canutility AA Divadip DD Die Funktion sortiert UnitSales von klein nach groß, auf der Suche nach dem Customer-Wert mit dem kleinsten UnitSales-Wert, der kleinsten Bestellung. Da sich CC auf die kleinste Bestellung bezieht (UnitSales= 2) für Kunde Astrida. AA entspricht der kleinsten Bestellung (4) für Kunde Betacab, AA entspricht der kleinsten Bestellung (8) für Kunde Canutility und DD der kleinsten Bestellung (10) für Kunde Divadip..
Vorgabe: Die Tabelle **Temp** wird wie im vorherigen Beispiel geladen: LOAD Customer,FirstSortedValue(Product, -UnitSales) as MyProductWithLargestOrderByCustomer Resident Temp Group By Customer;	Customer MyProductWithLargestOrderByCustomer Astrida AA Betacab DD Canutility CC Divadip - Dem sort_weight-Argument geht ein Minuszeichen voran, weshalb die Funktion zuerst die größten Werte sortiert. Da AA der größten Bestellung (Wert von UnitSales:18) für Kunde Astrida, DD der größten Bestellung (12) für Kunde Betacab und CC der größten Bestellung (13) für Kunde Canutility entspricht. Für die größte Bestellung (16) für Kunde Divadip gibt es zwei identische Werte, weshalb NULL ausgegeben wird.
Vorgabe: Die Tabelle **Temp** wird wie im vorherigen Beispiel geladen: LOAD Customer,FirstSortedValue(distinct Product, -UnitSales) as MyProductWithSmallestOrderByCustomer Resident Temp Group By Customer;	Customer MyProductWithLargestOrderByCustomer Astrida AA Betacab DD Canutility CC Divadip AA Dies entspricht dem vorigen Beispiel, außer dass der distinct-Zusatz verwendet wird. Das führt zu einer Ignorierung des duplizierten Ergebnisses für Divadip, wodurch ein Wert, der nicht gleich NULL ist, ausgegeben wird.

FirstSortedValue - Diagrammfunktion

FirstSortedValue() gibt den Wert der in **value** festgelegten Formel zurück, der dem Ergebnis der Sortierung des **sort_weight**-Arguments entspricht, z. B. der Name des Produkts mit dem niedrigsten Preis pro Einheit. Der n-te-Wert in der Sortierreihenfolge kann in **rank** festgelegt werden. Weist mehr als ein Ergebnis dasselbe Feld **sort_weight** für die festgelegte Funktion **rank** auf, gibt die Funktion NULL zurück.

Syntax:

```
FirstSortedValue([{SetExpression}] [DISTINCT] [TOTAL [<fld {,fld}>]] value,
sort_weight [,rank])
```

Rückgabe Datentyp: dual

Argumente:

Argumente

Argument	Beschreibung
value	Output-Feld. Die Funktion findet den Wert der Formel **value**, der dem Ergebnis der Sortierung von **sort_weight** entspricht.
sort_weight	Input-Feld. Die Formel mit den Daten, die sortiert werden sollen. Der erste (niedrigste) Wert von **sort_weight** wird gefunden, aus dem der entsprechende Wert der Formel **value** bestimmt wird. Wenn Sie ein Minuszeichen vor **sort_weight** voranstellen, liefert die Funktion stattdessen den letzten (höchsten) sortierten Wert.
rank	Durch **rank** "n" größer 1 wird der n-te Wert ausgegeben.
SetExpression	Standardmäßig berechnet sich die Aggregierungsfunktion über alle wählbaren Werte. Alternativ können Sie die der Berechnung zugrunde liegenden Werte über die Auswahlformel bestimmen.
DISTINCT	Der Zusatz **DISTINCT** vor den Funktionsargumenten bewirkt, dass bei der Auswertung der Funktionsargumente entstehende Duplikate nicht berücksichtigt werden.
TOTAL	Der Zusatz **TOTAL** vor der Funktion bewirkt, dass die Berechnung über alle ausgewählten bzw. wählbaren Werte erfolgt, und nicht nur über diejenigen, die zu dem Wert der aktuellen Dimension zählen. Die Dimensionen des Diagramms werden also nicht berücksichtigt. Mit **TOTAL [<fld {.fld}>]**, wobei auf den Zusatz **TOTAL** eine Liste aus mindestens einem Feldnamen (d. h. einer Teilmenge der Diagrammdimensionsvariablen) folgt, erstellen Sie eine Teilmenge aller möglichen Werte.

Beispiele und Ergebnisse:

Daten

Customer	Product	UnitSales	UnitPrice
Astrida	AA	4	16
Astrida	AA	10	15
Astrida	BB	9	9
Betacab	BB	5	10
Betacab	CC	2	20
Betacab	DD	-	25
Canutility	AA	8	15
Canutility	CC	-	19

Beispiele und Ergebnisse

Beispiel	Ergebnis
`firstsortedvalue (Product, UnitPrice)`	BB, da es das `Product` mit dem niedrigsten `UnitPrice` (9) ist.
`firstsortedvalue (Product, UnitPrice, 2)`	BB, da es das `Product` mit dem zweitniedrigsten `UnitPrice` (10) ist.
`firstsortedvalue (Customer, -UnitPrice, 2)`	Betacab, da es der `Customer` mit dem `Product` ist, das den zweithöchsten `UnitPrice` (20) aufweist.
`firstsortedvalue (Customer, UnitPrice, 3)`	NULL, da zwei Werte von `Customer` (Astrida und Canutility) mit demselben rank (drittniedrigster) `UnitPrice`(15) vorliegen. Verwenden Sie den Zusatz distinct, um sicherzustellen, dass keine unerwarteten NULL-Ergebnisse auftreten.
`firstsortedvalue (Customer, -UnitPrice*UnitSales, 2)`	Canutility, da dies der `Customer` mit dem zweithöchsten Bestellwert ist: `UnitPrice` multipliziert mit `UnitSales` (120).

In Beispielen verwendete Daten:

```
ProductData:
LOAD * inline [
Customer|Product|UnitSales|UnitPrice
Astrida|AA|4|16
Astrida|AA|10|15
Astrida|BB|9|9
Betacab|BB|5|10
Betacab|CC|2|20
Betacab|DD||25
Canutility|AA|8|15
```

```
Canutility|CC||19
] (delimiter is '|');
```

Max

Max() liefert den höchsten numerischen Wert der aggregierten Daten in der Formel, wie in einer Bedingung **group by** definiert wurde. Durch die Festlegung eines **rank** n kann der höchste n-te Wert gefunden werden.

Syntax:

```
Max ( expr [, rank])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
expr Expression	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.
rank Expression	Der Standardwert von **rank** ist 1, was dem höchsten Wert entspricht. Wird **rank** als 2 angegeben, wird der zweithöchste Wert ausgegeben. Wird **rank** als 3 angegeben, wird der dritthöchste Wert ausgegeben usw.

Beispiele und Ergebnisse:

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Fügen Sie einem Arbeitsblatt in unserer App dann mindestens diejenigen Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um das Ergebnis anzuzeigen.

Damit die Anzeige genauso wie in der unteren Ergebnisspalte aussieht, schalten Sie im Eigenschaftsfenster unter "Sortierung" von "Auto" auf "Benutzerdefiniert" um und heben Sie anschließend die numerische und alphabetische Sortierung auf.

Beispiel:

```
Temp:
LOAD * inline [
Customer|Product|OrderNumber|UnitSales|CustomerID
Astrida|AA|1|10|1
Astrida|AA|7|18|1
Astrida|BB|4|9|1
Astrida|CC|6|2|1
Betacab|AA|5|4|2
Betacab|BB|2|5|2
Betacab|DD
Canutility|DD|3|8
Canutility|CC
] (delimiter is '|');

Max:
LOAD Customer, Max(UnitSales) as MyMax Resident Temp Group By Customer;
```

Ergebnistabelle

Customer	MyMax
Astrida	18
Betacab	5
Canutility	8

Beispiel:

Vorgabe: Die Tabelle **Temp** wird wie im vorherigen Beispiel geladen:

```
LOAD Customer, Max(UnitSales,2) as MyMaxRank2 Resident Temp Group By Customer;
```

Ergebnistabelle

Customer	MyMaxRank2
Astrida	10
Betacab	4
Canutility	-

Max - Diagrammfunktion

Max() liefert den höchsten Wert der aggregierten Daten. Durch die Festlegung eines **rank** n kann der höchste n-te Wert gefunden werden.

*Es ist auch ratsam, sich **FirstSortedValue** und **rangemax** anzusehen. Diese haben eine ähnliche Funktion wie die Funktion **Max**.*

Syntax:

```
Max([{SetExpression}] [TOTAL [<fld {,fld}>]] expr [,rank])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
expr	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.
rank	Der Standardwert von **rank** ist 1, was dem höchsten Wert entspricht. Wird **rank** als 2 angegeben, wird der zweithöchste Wert ausgegeben. Wird **rank** als 3 angegeben, wird der dritthöchste Wert ausgegeben usw.
SetExpression	Standardmäßig berechnet sich die Aggregierungsfunktion über alle wählbaren Werte. Alternativ können Sie die der Berechnung zugrunde liegenden Werte über die Auswahlformel bestimmen.

Argument	Beschreibung
TOTAL	Der Zusatz **TOTAL** vor der Funktion bewirkt, dass die Berechnung über alle ausgewählten bzw. wählbaren Werte erfolgt, und nicht nur über diejenigen, die zu dem Wert der aktuellen Dimension zählen. Die Dimensionen des Diagramms werden also nicht berücksichtigt. Mit **TOTAL [<fld {.fld}>]**, wobei auf den Zusatz **TOTAL** eine Liste aus mindestens einem Feldnamen (d. h. einer Teilmenge der Diagrammdimensionsvariablen) folgt, erstellen Sie eine Teilmenge aller möglichen Werte.

Beispiele und Ergebnisse:

Daten

Customer	Product	UnitSales	UnitPrice
Astrida	AA	4	16
Astrida	AA	10	15
Astrida	BB	9	9
Betacab	BB	5	10
Betacab	CC	2	20
Betacab	DD	-	25
Canutility	AA	8	15
Canutility	CC	-	19

Beispiele und Ergebnisse

Beispiele	Ergebnisse
`Max(UnitSales)`	10, da dies der höchste Wert in `UnitSales` ist.
Der Wert einer Bestellung wird aus der Anzahl der verkauften Einheiten (`UnitSales`) multipliziert mit dem Stückpreis berechnet. `Max (UnitSales*UnitPrice)`	150, da dies der höchste Wert der Berechnung aus allen wählbaren Werten von (`UnitSales`)*(`UnitPrice`) ist.
`Max(UnitSales, 2)`	9, da dies der zweithöchste Wert ist.

Beispiele	Ergebnisse
`Max(TOTAL UnitSales)`	10, da der Zusatz TOTAL bedeutet, dass der höchste wählbare Wert gefunden wird, ohne die Dimensionen des Diagramms zu berücksichtigen. Bei einem Diagramm mit Customer als Dimension sorgt der Zusatz TOTAL dafür, dass der maximale Wert übergreifend über den vollständigen Datensatz anstelle des Maximalwerts UnitSales für jeden Kunden zurückgegeben wird.
Treffen Sie die Auswahl `Customer B.` `Max({1} TOTAL UnitSales)`	10, unabhängig von der getroffenen Auswahl, da die Set Analysis-Formel {1} den Datensatz so definiert, dass er als ALL bewertet wird, unabhängig davon, welche Auswahl getroffen wird.

In Beispielen verwendete Daten:

```
ProductData:
LOAD * inline [
Customer|Product|UnitSales|UnitPrice
Astrida|AA|4|16
Astrida|AA|10|15
Astrida|BB|9|9
Betacab|BB|5|10
Betacab|CC|2|20
Betacab|DD||25
Canutility|AA|8|15
Canutility|CC||19
] (delimiter is '|');
```

Siehe auch:

- *FirstSortedValue - Diagrammfunktion (page 347)*
- *RangeMax (page 1375)*

Min

Min() liefert den niedrigsten numerischen Wert der aggregierten Daten in der Formel, wie in einer Bedingung **group by** definiert wurde. Durch die Festlegung eines **rank** n kann der niedrigste n-te Wert gefunden werden.

Syntax:

```
Min ( expr [, rank])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
expr Expression	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.
rank Expression	Der Standardwert von **rank** ist 1, was dem niedrigsten Wert entspricht. Durch die Angabe von **rank** als 2 wird der zweitniedrigste Wert zurückgegeben. Falls **rank** 3 ist, wird der drittniedrigste Wert zurückgegeben und so weiter.

Beispiele und Ergebnisse:

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Fügen Sie einem Arbeitsblatt in unserer App dann mindestens diejenigen Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um das Ergebnis anzuzeigen.

Damit die Anzeige genauso wie in der unteren Ergebnisspalte aussieht, schalten Sie im Eigenschaftsfenster unter "Sortierung" von "Auto" auf "Benutzerdefiniert" um und heben Sie anschließend die numerische und alphabetische Sortierung auf.

Beispiel:

```
Temp:
LOAD * inline [
Customer|Product|OrderNumber|UnitSales|CustomerID
Astrida|AA|1|10|1
Astrida|AA|7|18|1
Astrida|BB|4|9|1
Astrida|CC|6|2|1
Betacab|AA|5|4|2
Betacab|BB|2|5|2
Betacab|DD
Canutility|DD|3|8
Canutility|CC
] (delimiter is '|');
Min:
LOAD Customer, Min(UnitSales) as MyMin Resident Temp Group By Customer;
```

Ergebnistabelle

Customer	MyMin
Astrida	2
Betacab	4
Canutility	8

Beispiel:

Vorgabe: Die Tabelle **Temp** wird wie im vorherigen Beispiel geladen:

```
LOAD Customer, Min(UnitSales,2) as MyMinRank2 Resident Temp Group By Customer;
```

Ergebnistabelle

Customer	MyMinRank2
Astrida	9
Betacab	5
Canutility	-

Min - Diagrammfunktion

Min() liefert den niedrigsten Wert der aggregierten Daten. Durch die Festlegung eines **rank** n kann der niedrigste n-te Wert gefunden werden.

*Es ist auch ratsam, sich **FirstSortedValue** und **rangemin** anzusehen. Diese haben eine ähnliche Funktion wie die Funktion **Min**.*

Syntax:

```
Min({[SetExpression] [TOTAL [<fld {,fld}>]]} expr [,rank])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
expr	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.
rank	Der Standardwert von **rank** ist 1, was dem niedrigsten Wert entspricht. Durch die Angabe von **rank** als 2 wird der zweitniedrigste Wert zurückgegeben. Falls **rank** 3 ist, wird der drittniedrigste Wert zurückgegeben und so weiter.
SetExpression	Standardmäßig berechnet sich die Aggregierungsfunktion über alle wählbaren Werte. Alternativ können Sie die der Berechnung zugrunde liegenden Werte über die Auswahlformel bestimmen.
TOTAL	Der Zusatz **TOTAL** vor der Funktion bewirkt, dass die Berechnung über alle ausgewählten bzw. wählbaren Werte erfolgt, und nicht nur über diejenigen, die zu dem Wert der aktuellen Dimension zählen. Die Dimensionen des Diagramms werden also nicht berücksichtigt. Mit **TOTAL [<fld {.fld}>]**, wobei auf den Zusatz **TOTAL** eine Liste aus mindestens einem Feldnamen (d. h. einer Teilmenge der Diagrammdimensionsvariablen) folgt, erstellen Sie eine Teilmenge aller möglichen Werte.

Beispiele und Ergebnisse:

Daten

Customer	Product	UnitSales	UnitPrice
Astrida	AA	4	16
Astrida	AA	10	15
Astrida	BB	9	9
Betacab	BB	5	10
Betacab	CC	2	20
Betacab	DD	-	25
Canutility	AA	8	15
Canutility	CC	-	19

Die Funktion Min() muss einen Nicht-NULL-Wert aus der Reihe von Werten zurückgeben, die durch die Formel bestimmt werden, falls eine existiert. Da in den Beispielen also NULL-Werte in den Daten enthalten sind, liefert die Funktion den ersten Nicht-NULL-Wert durch die Auswertung der Formel.

Beispiele und Ergebnisse

Beispiele	Ergebnisse
`Min(UnitSales)`	2, da dies der niedrigste Nicht-NULL-Wert in `UnitSales` ist.
Der Wert einer Bestellung wird aus der Anzahl der verkauften Einheiten (`UnitSales`) multipliziert mit dem Stückpreis berechnet. `Min (UnitSales*UnitPrice)`	40, da dies der niedrigste Nicht-NULL-Wert ist, der sich aus der Berechnung aller möglichen Werte von (`UnitSales`)*(`UnitPrice`) ergibt.
`Min(UnitSales, 2)`	4, da dies der zweitniedrigste Wert (nach den NULL-Werten) ist.
`Min(TOTAL UnitSales)`	2, da der Zusatz TOTAL bedeutet, dass der niedrigste wählbare Wert gefunden wird, ohne die Dimensionen des Diagramms zu berücksichtigen. Bei einem Diagramm mit der Dimension Customer sorgt der Zusatz TOTAL dafür, dass der minimale Wert übergreifend über den vollständigen Datensatz anstelle des Minimalwerts UnitSales für jeden Kunden zurückgegeben wird.

Beispiele	Ergebnisse

Treffen Sie die Auswahl Customer B.
 	
`Min({1}`
`TOTAL UnitSales)`
	2, da dies unabhängig von der Auswahl von Customer B ist.
	
	Die Set Analysis-Formel {1} definiert den Datensatz so, dass er als ALL bewertet wird, unabhängig davon, welche Auswahl getroffen wird.

In Beispielen verwendete Daten:

```
ProductData:
LOAD * inline [
Customer|Product|UnitSales|UnitPrice
Astrida|AA|4|16
Astrida|AA|10|15
Astrida|BB|9|9
Betacab|BB|5|10
Betacab|CC|2|20
Betacab|DD||25
Canutility|AA|8|15
Canutility|CC||19
] (delimiter is '|');
```

Siehe auch:

- *FirstSortedValue - Diagrammfunktion (page 347)*
- *RangeMin (page 1379)*

Mode

Mode() liefert den am häufigsten vorkommenden Wert, also den Moduswert der aggregierten Daten in der Formel, wie in einer Bedingung **group by** definiert wurde. Die Funktion **Mode()** kann sowohl numerische Werte als auch Textwerte liefern.

Syntax:

```
Mode ( expr)
```

Rückgabe Datentyp: dual

Argumente

Argument	Beschreibung
expr Expression	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.

Beschränkungen:

Kommen mehrere Werte gleich oft vor, wird NULL ausgegeben.

Beispiele und Ergebnisse:

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Fügen Sie einem Arbeitsblatt in unserer App dann mindestens diejenigen Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um das Ergebnis anzuzeigen.

Damit die Anzeige genauso wie in der unteren Ergebnisspalte aussieht, schalten Sie im Eigenschaftsfenster unter "Sortierung" von "Auto" auf "Benutzerdefiniert" um und heben Sie anschließend die numerische und alphabetische Sortierung auf.

Skriptbeispiele

Beispiel	Ergebnis

`Temp:`
`LOAD * inline [`
`Customer|Product|OrderNumber|UnitSales|CustomerID`
`Astrida|AA|1|10|1`
`Astrida|AA|7|18|1`
`Astrida|BB|4|9|1`
`Astrida|CC|6|2|1`
`Betacab|AA|5|4|2`
`Betacab|BB|2|5|2`
`Betacab|DD`
`Canutility|DD|3|8`
`Canutility|CC`
`] (delimiter is '|');`
 	
`Mode:`
`LOAD Customer, Mode(Product) as MyMostOftenSoldProduct`
`Resident Temp Group By Customer;`
	MyMostOftenSoldProduct
	
	AA
	
	da AA das einzige Produkt ist, das mehrmals verkauft wurde.

Mode - Diagrammfunktion

Mode() liefert den am häufigsten vorkommenden Wert, also den Moduswert, in den aggregierten Daten. Die Funktion **Mode()** kann sowohl Textwerte als auch numerische Werte verarbeiten.

Syntax:

```
Mode({[SetExpression] [TOTAL [<fld {,fld}>]]} expr)
```

Rückgabe Datentyp: dual

Argumente:

Argumente

Argument	Beschreibung
expr	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.
SetExpression	Standardmäßig berechnet sich die Aggregierungsfunktion über alle wählbaren Werte. Alternativ können Sie die der Berechnung zugrunde liegenden Werte über die Auswahlformel bestimmen.

Argument	Beschreibung
TOTAL	Der Zusatz **TOTAL** vor der Funktion bewirkt, dass die Berechnung über alle ausgewählten bzw. wählbaren Werte erfolgt, und nicht nur über diejenigen, die zu dem Wert der aktuellen Dimension zählen. Die Dimensionen des Diagramms werden also nicht berücksichtigt. Mit **TOTAL [<fld {.fld}>]**, wobei auf den Zusatz **TOTAL** eine Liste aus mindestens einem Feldnamen (d. h. einer Teilmenge der Diagrammdimensionsvariablen) folgt, erstellen Sie eine Teilmenge aller möglichen Werte.

Beispiele und Ergebnisse:

Daten

Customer	Product	UnitSales	UnitPrice
Astrida	AA	4	16
Astrida	AA	10	15
Astrida	BB	9	9
Betacab	BB	5	10
Betacab	CC	2	20
Betacab	DD	-	25
Canutility	AA	8	15
Canutility	CC	-	19

Beispiele und Ergebnisse

Beispiele	Ergebnisse
`Mode(UnitPrice)` Treffen Sie die Auswahl `Customer A`.	15, da es der am häufigsten vorkommende Wert in `UnitSales` ist. Liefert NULL (-). Kein einzelner Wert kommt häufiger als ein anderer vor.
`Mode(Product)` Nehmen Sie die Auswahl `Customer A` vor	AA, da es der am häufigsten vorkommende Wert in `Product` ist. Liefert NULL (-). Kein einzelner Wert kommt häufiger als ein anderer vor.
`Mode (TOTAL UnitPrice)`	15, da der Zusatz TOTAL bedeutet, dass der am häufigsten vorkommende Wert 15 ist, ohne die Dimensionen des Diagramms zu berücksichtigen.

Beispiele	Ergebnisse
Nehmen Sie die Auswahl `Customer B` vor. `Mode({1} TOTAL UnitPrice)`	15, unabhängig von der getroffenen Auswahl, da die Set Analysis-Formel {1} den Datensatz so definiert, dass er als ALL bewertet wird, unabhängig davon, welche Auswahl getroffen wird.

In Beispielen verwendete Daten:

```
ProductData:
LOAD * inline [
Customer|Product|UnitSales|UnitPrice
Astrida|AA|4|16
Astrida|AA|10|15
Astrida|BB|9|9
Betacab|BB|5|10
Betacab|CC|2|20
Betacab|DD||25
Canutility|AA|8|15
Canutility|CC||19
] (delimiter is '|');
```

Siehe auch:

- *Avg - Diagrammfunktion (page 419)*
- *Median - Diagrammfunktion (page 458)*

Only

Only() liefert einen Wert, wenn die aggregierten Daten nur ein einziges mögliches Ergebnis liefern. Enthält ein Datensatz nur einen einzigen Wert, wird dieser Wert geliefert, ansonsten NULL. Verwenden Sie die Bedingung **group by** für die Evaluierung über mehrere Datensätze. Die Funktion **Only()** kann numerische und Textwerte liefern.

Syntax:

```
Only ( expr )
```

Rückgabe Datentyp: dual

Argumente

Argument	Beschreibung
expr Expression	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.

Beispiele und Ergebnisse:

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Fügen Sie einem Arbeitsblatt in unserer App dann mindestens diejenigen Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um das Ergebnis anzuzeigen.

Damit die Anzeige genauso wie in der unteren Ergebnisspalte aussieht, schalten Sie im Eigenschaftsfenster unter "Sortierung" von "Auto" auf "Benutzerdefiniert" um und heben Sie anschließend die numerische und alphabetische Sortierung auf.

```
Temp:
LOAD * inline [
Customer|Product|OrderNumber|UnitSales|CustomerID
Astrida|AA|1|10|1
Astrida|AA|7|18|1
Astrida|BB|4|9|1
Astrida|CC|6|2|1
Betacab|AA|5|4|2
Betacab|BB|2|5|2
Betacab|DD
Canutility|DD|3|8
Canutility|CC
] (delimiter is '|');
Only:
LOAD Customer, Only(CustomerID) as MyUniqIDCheck Resident Temp Group By Customer;
```

Ergebnistabelle

Customer	MyUniqIDCheck
Astrida	1 da nur Kunde Astrida vollständige Datensätze einschließlich CustomerID aufweist.

Only - Diagrammfunktion

Only() liefert einen Wert, wenn die aggregierten Daten nur ein einziges mögliches Ergebnis liefern. Wenn Sie beispielsweise nach dem einzigen Produkt suchen, bei dem der Preis pro Einheit 9 beträgt, wird NULL geliefert, wenn mehr als ein Produkt einen Preis pro Einheit von 9 aufweist.

Syntax:

```
Only([{SetExpression}] [TOTAL [<fld {,fld}>]] expr)
```

Rückgabe Datentyp: dual

Argumente:

Argumente

Argument	Beschreibung
expr	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.
SetExpression	Standardmäßig berechnet sich die Aggregierungsfunktion über alle wählbaren Werte. Alternativ können Sie die der Berechnung zugrunde liegenden Werte über die Auswahlformel bestimmen.

Argument	Beschreibung
TOTAL	Der Zusatz **TOTAL** vor der Funktion bewirkt, dass die Berechnung über alle ausgewählten bzw. wählbaren Werte erfolgt, und nicht nur über diejenigen, die zu dem Wert der aktuellen Dimension zählen. Die Dimensionen des Diagramms werden also nicht berücksichtigt. Mit **TOTAL [<fld {.fld}>]**, wobei auf den Zusatz **TOTAL** eine Liste aus mindestens einem Feldnamen (d. h. einer Teilmenge der Diagrammdimensionsvariablen) folgt, erstellen Sie eine Teilmenge aller möglichen Werte.

Verwenden Sie Only(), wenn Sie in den Beispieldaten bei mehreren möglichen Werten ein NULL-Ergebnis wünschen.

Beispiele und Ergebnisse:

Daten

Customer	Product	UnitSales	UnitPrice
Astrida	AA	4	16
Astrida	AA	10	15
Astrida	BB	9	9
Betacab	BB	5	10
Betacab	CC	2	20
Betacab	DD	-	25
Canutility	AA	8	15
Canutility	CC	-	19

Beispiele und Ergebnisse

Beispiele	Ergebnisse
`Only({<UnitPrice={9}>} Product)`	BB, da es das einzige `Product` ist, das einen `UnitPrice` von '9' aufweist.
`Only({<Product={DD}>} Customer)`	Betacab, da es der einzige `Customer` ist, der ein `Product` namens 'DD' verkauft.
`Only({<UnitPrice={20}>} UnitSales)`	Die Anzahl an `UnitSales`, bei der `UnitPrice`20 ist, ergibt 2, da es nur einen Wert von `UnitSales` gibt, bei dem der `UnitPrice` 20 ist.
`Only({<UnitPrice={15}>} UnitSales)`	NULL, da es zwei Werte von `UnitSales` gibt, bei denen der `UnitPrice` 15 ist.

In Beispielen verwendete Daten:

```
ProductData:
LOAD * inline [
Customer|Product|UnitSales|UnitPrice
Astrida|AA|4|16
Astrida|AA|10|15
Astrida|BB|9|9
Betacab|BB|5|10
Betacab|CC|2|20
Betacab|DD||25
Canutility|AA|8|15
Canutility|CC||19
] (delimiter is '|');
```

Sum

Sum() berechnet den Gesamtwert der in der Formel aggregierten Werte, wie in einer Bedingung **group by** definiert wurde.

Syntax:

```
sum ( [ distinct] expr)
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
distinct	Ist der Formel das Wort **distinct** vorangestellt, werden Dubletten nicht berücksichtigt.
expr Expression	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.

Beispiele und Ergebnisse:

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Fügen Sie einem Arbeitsblatt in unserer App dann mindestens diejenigen Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um das Ergebnis anzuzeigen.

Damit die Anzeige genauso wie in der unteren Ergebnisspalte aussieht, schalten Sie im Eigenschaftsfenster unter "Sortierung" von "Auto" auf "Benutzerdefiniert" um und heben Sie anschließend die numerische und alphabetische Sortierung auf.

```
Temp:
LOAD * inline [
Customer|Product|OrderNumber|UnitSales|CustomerID
Astrida|AA|1|10|1
Astrida|AA|7|18|1
Astrida|BB|4|9|1
Astrida|CC|6|2|1
Betacab|AA|5|4|2
Betacab|BB|2|5|2
Betacab|DD
Canutility|DD|3|8
Canutility|CC
] (delimiter is '|');
```

```
Sum:
LOAD Customer, Sum(UnitSales) as MySum Resident Temp Group By Customer;
```

Ergebnistabelle

Customer	MySum
Astrida	39
Betacab	9
Canutility	8

Sum - Diagrammfunktion

Sum() berechnet die Gesamtsumme der von der Formel vorgegebenen Werte oder Felder über die aggregierten Daten.

Syntax:

```
Sum([{SetExpression}] [DISTINCT] [TOTAL [<fld {,fld}>]] expr)
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
expr	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.
SetExpression	Standardmäßig berechnet sich die Aggregierungsfunktion über alle wählbaren Werte. Alternativ können Sie die der Berechnung zugrunde liegenden Werte über die Auswahlformel bestimmen.
DISTINCT	Der Zusatz **DISTINCT** vor den Funktionsargumenten bewirkt, dass bei der Auswertung der Funktionsargumente entstehende Duplikate nicht berücksichtigt werden. *Obwohl der Zusatz DISTINCT unterstützt wird, sollte er nur mit äußerster Vorsicht eingesetzt werden. Denn dies kann für den Benutzer irreführend sein und dieser denkt möglicherweise, dass ein Gesamtwert angezeigt wird, obwohl bestimmte Daten ausgelassen wurden.*
TOTAL	Der Zusatz **TOTAL** vor der Funktion bewirkt, dass die Berechnung über alle ausgewählten bzw. wählbaren Werte erfolgt, und nicht nur über diejenigen, die zu dem Wert der aktuellen Dimension zählen. Die Dimensionen des Diagramms werden also nicht berücksichtigt. Mit **TOTAL [<fld {.fld}>]**, wobei auf den Zusatz **TOTAL** eine Liste aus mindestens einem Feldnamen (d. h. einer Teilmenge der Diagrammdimensionsvariablen) folgt, erstellen Sie eine Teilmenge aller möglichen Werte.

Beispiele und Ergebnisse:

Daten

Customer	Product	UnitSales	UnitPrice
Astrida	AA	4	16
Astrida	AA	10	15
Astrida	BB	9	9
Betacab	BB	5	10
Betacab	CC	2	20
Betacab	DD	-	25
Canutility	AA	8	15
Canutility	CC	-	19

Beispiele und Ergebnisse

Beispiele	Ergebnisse
`Sum(UnitSales)`	38. Die Summe der Werte in `UnitSales`.
`Sum(UnitSales*UnitPrice)`	505. Die Summe von `UnitPrice` multipliziert mit `UnitSales` aggregiert.
`Sum (TOTAL UnitSales*UnitPrice)`	505 für alle Zeilen in der Tabelle sowie der Gesamtsumme, weil der Zusatz TOTAL bedeutet, dass die Summe immer noch 505 beträgt und die Dimensionen des Diagramms nicht berücksichtigt werden.
Nehmen Sie die Auswahl `Customer B` vor. `Sum({1} TOTAL UnitSales*UnitPrice)`	505, unabhängig von der getroffenen Auswahl, da die Set Analysis-Formel {1} den Datensatz so definiert, dass er als ALL bewertet wird, unabhängig davon, welche Auswahl getroffen wird.

In Beispielen verwendete Daten:

```
ProductData:
LOAD * inline [
Customer|Product|UnitSales|UnitPrice
Astrida|AA|4|16
Astrida|AA|10|15
Astrida|BB|9|9
Betacab|BB|5|10
Betacab|CC|2|20
Betacab|DD||25
Canutility|AA|8|15
Canutility|CC||19
] (delimiter is '|');
```

Aggregierung von Häufigkeiten

Funktionen zur Aggregierung von Häufigkeiten geben verschiedene Arten von Häufigkeiten einer Formel in Hinblick auf mehrere Datensätze in einem Datenladeskript bzw. eine Reihe von Werten in der Dimension eines Diagramms zurück.

Jede Funktion wird nach der Übersicht genauer beschrieben. Sie können auch auf den Funktionsnamen in der Syntax klicken, um direkt auf die Details zu der spezifischen Funktion zuzugreifen.

Funktionen zur Aggregierung von Häufigkeiten im Datenladeskript verwenden

Count

Count() liefert die Anzahl der in einer Formel aggregierten Werte, wie in einer Bedingung **group by** definiert wurde.

```
Count ([distinct ] expression | * )
```

MissingCount

MissingCount() liefert die Anzahl der in der Formel fehlenden aggregierten Werte, wie in einer Bedingung **group by** definiert wurde.

```
MissingCount ([ distinct ] expression)
```

NullCount

NullCount() liefert die Anzahl der in der Formel aggregierten NULL-Werte, wie in einer Bedingung **group by** definiert wurde.

```
NullCount ([ distinct ] expression)
```

NumericCount

NumericCount() liefert die Anzahl der in der Formel gefundenen numerischen Werte, wie in einer Bedingung **group by** definiert wurde.

```
NumericCount ([ distinct ] expression)
```

TextCount

TextCount() liefert die Anzahl der Feldwerte, die in der Formel nicht numerisch aggregiert wurden, wie in einer **group by**-Bedingung definiert wurde.

```
TextCount ([ distinct ] expression)
```

Funktionen zur Aggregierung von Häufigkeiten in Diagrammformeln verwenden

Die folgenden Aggregierungsfunktionen für Häufigkeiten können in Diagrammen verwendet werden.

Count

Count() aggregiert die Anzahl der Text- und numerischen Werte nach den Dimensionen des Diagramms.

```
Count - Diagrammfunktion({[SetExpression] [DISTINCT] [TOTAL [<fld {,fld}>]]}
expr)
```

MissingCount

MissingCount() aggregiert die Anzahl der fehlenden Werte nach den Dimensionen des Diagramms. Fehlende Werte sind alle nicht-numerischen Werte.

```
MissingCount - Diagrammfunktion({[SetExpression] [DISTINCT] [TOTAL [<fld
{,fld}>]] expr)
```

NullCount

NullCount() aggregiert die Anzahl der NULL-Werte nach den Dimensionen des Diagramms.

```
NullCount - Diagrammfunktion({[SetExpression][DISTINCT] [TOTAL [<fld
{,fld}>]]} expr)
```

NumericCount

NumericCount() aggregiert die Anzahl der numerischen Werte nach den Dimensionen des Diagramms.

```
NumericCount - Diagrammfunktion({[SetExpression] [DISTINCT] [TOTAL [<fld
{,fld}>]]} expr)
```

TextCount

TextCount() aggregiert die Anzahl der nicht numerischen Feldwerte nach den Dimensionen des Diagramms.

```
TextCount - Diagrammfunktion({[SetExpression] [DISTINCT] [TOTAL [<fld
{,fld}>]]} expr)
```

Count

Count() liefert die Anzahl der in einer Formel aggregierten Werte, wie in einer Bedingung **group by** definiert wurde.

Syntax:

```
Count( [distinct ] expr)
```

Rückgabe Datentyp: ganze Zahl

Argumente:

Argumente

Argument	Beschreibung
expr	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.
distinct	Ist der Formel das Wort **distinct** vorangestellt, werden Dubletten nicht berücksichtigt.

Beispiele und Ergebnisse:

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Fügen Sie einem Arbeitsblatt in unserer App dann mindestens diejenigen Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um das Ergebnis anzuzeigen.

Damit die Anzeige genauso wie in der unteren Ergebnisspalte aussieht, schalten Sie im Eigenschaftsfenster unter "Sortierung" von "Auto" auf "Benutzerdefiniert" um und heben Sie anschließend die numerische und alphabetische Sortierung auf.

Skriptbeispiele

Beispiel	**Ergebnis**

`Temp:`
`LOAD * inline [`
`Customer|Product|OrderNumber|UnitSales|UnitPrice`
`Astrida|AA|1|4|16`
`Astrida|AA|7|10|15`
`Astrida|BB|4|9|9`
`Betacab|CC|6|5|10`
`Betacab|AA|5|2|20`
`Betacab|BB|1|25| 25`
`Canutility|AA|3|8|15`
`Canutility|CC|||19`
`Divadip|CC|2|4|16`
`Divadip|DD|3|1|25`
`] (delimiter is '|');`
 	
`Count1:`
 	
`LOAD Customer,Count(OrderNumber) as OrdersByCustomer Resident Temp Group By Customer;`
	`Customer OrdersByCustomer`
	
	`Astrida 3`
	
	`Betacab 3`
	
	`Canutility 2`
	
	`Divadip 2`
	Sofern die Dimension Customer in der Tabelle im Arbeitsblatt enthalten ist, andernfalls lautet das Ergebnis für OrdersByCustomer 3, 2.

Vorgabe: Die Tabelle **Temp** wird wie im vorherigen Beispiel geladen:
 	
`LOAD Count(OrderNumber) as TotalOrderNumber Resident Temp;`
	`TotalOrderNumber`
	
	`10`

Vorgabe: Die Tabelle **Temp** wird wie im ersten Beispiel geladen:
 	
`LOAD Count(distinct OrderNumber) as TotalOrderNumber Resident Temp;`
	`TotalOrderNumber`
	
	`8`
	Denn zwei Werte von OrderNumber liegen mit demselben Wert 1 vor, und es gibt einen Nullwert.

Count - Diagrammfunktion

Count() aggregiert die Anzahl der Text- und numerischen Werte nach den Dimensionen des Diagramms.

Syntax:

```
Count({[SetExpression] [DISTINCT] [TOTAL [<fld {,fld}>]]} expr)
```

Rückgabe Datentyp: ganze Zahl

Argumente:

Argumente

Argument	Beschreibung
expr	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.
SetExpression	Standardmäßig berechnet sich die Aggregierungsfunktion über alle wählbaren Werte. Alternativ können Sie die der Berechnung zugrunde liegenden Werte über die Auswahlformel bestimmen.
DISTINCT	Der Zusatz **DISTINCT** vor den Funktionsargumenten bewirkt, dass bei der Auswertung der Funktionsargumente entstehende Duplikate nicht berücksichtigt werden.
TOTAL	Der Zusatz **TOTAL** vor der Funktion bewirkt, dass die Berechnung über alle ausgewählten bzw. wählbaren Werte erfolgt, und nicht nur über diejenigen, die zu dem Wert der aktuellen Dimension zählen. Die Dimensionen des Diagramms werden also nicht berücksichtigt. Mit **TOTAL [<fld {.fld}>]**, wobei auf den Zusatz **TOTAL** eine Liste aus mindestens einem Feldnamen (d. h. einer Teilmenge der Diagrammdimensionsvariablen) folgt, erstellen Sie eine Teilmenge aller möglichen Werte.

Beispiele und Ergebnisse:

Data

Customer	Product	OrderNumber	UnitSales	Unit Price
Astrida	AA	1	4	16
Astrida	AA	7	10	15
Astrida	BB	4	9	9
Betacab	BB	6	5	10
Betacab	CC	5	2	20
Betacab	DD	1	25	25
Canutility	AA	3	8	15
Canutility	CC			19
Divadip	AA	2	4	16
Divadip	DD	3		25

Für die folgenden Beispiele wird vorausgesetzt, dass alle Kunden ausgewählt sind, sofern nicht anders angegeben.

Beispiele und Ergebnisse

Beispiel	**Ergebnis**
`Count(OrderNumber)`	10, da es 10 Felder gibt, die einen Wert für OrderNumber haben könnten. Alle Datensätze einschließlich der leeren werden gezählt. *"0" zählt als Wert und nicht als leere Zelle. Wenn jedoch eine Kennzahl für eine Dimension auf 0 aggregiert wird, wird diese Dimension nicht in Diagramme eingeschlossen.*
`Count(Customer)`	10, da Count die Anzahl der Vorkommnisse in allen Feldern auswertet.
`Count(DISTINCT [Customer])`	4, da mithilfe des Qualifizierers Distinct Count nur eindeutige Vorkommnisse auswertet.
Vorgabe: Kunde Canutility ist ausgewählt `Count (OrderNumber)/Count ({1} TOTAL OrderNumber)`	0,2, da die Formel die Anzahl der Bestellungen des ausgewählten Kunden als Prozentsatz der Bestellungen aller Kunden liefert. In diesem Fall 2 / 10.
Wenn Kunden Astrida und Canutility ausgewählt sind `Count(TOTAL <Product> OrderNumber)`	5, da das die Anzahl der Bestellungen für Produkte von den ausgewählten Kunden ist und leere Zellen mitgezählt werden.

In Beispielen verwendete Daten:

```
Temp:
LOAD * inline [
Customer|Product|OrderNumber|UnitSales|UnitPrice
Astrida|AA|1|4|16
Astrida|AA|7|10|15
Astrida|BB|4|9|9
Betacab|CC|6|5|10
Betacab|AA|5|2|20
Betacab|BB|1|25| 25
Canutility|AA|3|8|15
Canutility|CC|||19
Divadip|CC|2|4|16
Divadip|DD|3|1|25
] (delimiter is '|');
```

MissingCount

MissingCount() liefert die Anzahl der in der Formel fehlenden aggregierten Werte, wie in einer Bedingung **group by** definiert wurde.

Syntax:

```
MissingCount ( [ distinct ] expr)
```

Rückgabe Datentyp: ganze Zahl

Argumente:

Argumente

Argument	Beschreibung
expr Expression	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.
distinct	Ist der Formel das Wort **distinct** vorangestellt, werden Dubletten nicht berücksichtigt.

Beispiele und Ergebnisse:

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Fügen Sie einem Arbeitsblatt in unserer App dann mindestens diejenigen Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um das Ergebnis anzuzeigen.

Damit die Anzeige genauso wie in der unteren Ergebnisspalte aussieht, schalten Sie im Eigenschaftsfenster unter "Sortierung" von "Auto" auf "Benutzerdefiniert" um und heben Sie anschließend die numerische und alphabetische Sortierung auf.

Skriptbeispiele

Beispiel	Ergebnis
Temp: LOAD * inline [Customer|Product|OrderNumber|UnitSales|UnitPrice Astrida|AA|1|4|16 Astrida|AA|7|10|15 Astrida|BB|4|9|9 Betacab|CC|6|5|10 Betacab|AA|5|2|20 Betacab|BB||| 25 Canutility|AA|||15 Canutility|CC| ||19 Divadip|CC|2|4|16 Divadip|DD|3|1|25] (delimiter is '|'); MissCount1: LOAD Customer,MissingCount(OrderNumber) as MissingOrdersByCustomer Resident Temp Group By Customer; Load MissingCount(OrderNumber) as TotalMissingCount Resident Temp;	Customer MissingOrdersByCustomer Astrida 0 Betacab 1 Canutility 2 Divadip 0 Der zweite Befehl ergibt: TotalMissingCount 3 in einer Tabelle mit dieser Dimension.
Vorgabe: Die Tabelle **Temp** wird wie im vorherigen Beispiel geladen: LOAD MissingCount(distinct OrderNumber) as TotalMissingCountDistinct Resident Temp;	TotalMissingCountDistinct 1 Da nur ein OrderNumber-Wert fehlt.

MissingCount - Diagrammfunktion

MissingCount() aggregiert die Anzahl der fehlenden Werte nach den Dimensionen des Diagramms. Fehlende Werte sind alle nicht-numerischen Werte.

Syntax:

MissingCount({[SetExpression] [**DISTINCT**] [**TOTAL** [<fld {,fld}>]]} expr**)**

Rückgabe Datentyp: ganze Zahl

Argumente:

Argumente

Argument	Beschreibung
expr	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.
SetExpression	Standardmäßig berechnet sich die Aggregierungsfunktion über alle wählbaren Werte. Alternativ können Sie die der Berechnung zugrunde liegenden Werte über die Auswahlformel bestimmen.

Argument	Beschreibung
DISTINCT	Der Zusatz **DISTINCT** vor den Funktionsargumenten bewirkt, dass bei der Auswertung der Funktionsargumente entstehende Duplikate nicht berücksichtigt werden.
TOTAL	Der Zusatz **TOTAL** vor der Funktion bewirkt, dass die Berechnung über alle ausgewählten bzw. wählbaren Werte erfolgt, und nicht nur über diejenigen, die zu dem Wert der aktuellen Dimension zählen. Die Dimensionen des Diagramms werden also nicht berücksichtigt. Mit **TOTAL [<fld {.fld}>]**, wobei auf den Zusatz **TOTAL** eine Liste aus mindestens einem Feldnamen (d. h. einer Teilmenge der Diagrammdimensionsvariablen) folgt, erstellen Sie eine Teilmenge aller möglichen Werte.

Beispiele und Ergebnisse:

Data

Customer	Product	OrderNumber	UnitSales	Unit Price
Astrida	AA	1	4	16
Astrida	AA	7	10	15
Astrida	BB	4	9	9
Betacab	BB	6	5	10
Betacab	CC	5	2	20
Betacab	DD			25
Canutility	AA			15
Canutility	CC			19
Divadip	AA	2	4	16
Divadip	DD	3		25

Beispiele und Ergebnisse

Beispiel	Ergebnis
`MissingCount([OrderNumber])`	3, da 3 der 10 Felder OrderNumber leer sind *"0" zählt als Wert und nicht als leere Zelle. Wenn jedoch eine Kennzahl für eine Dimension auf 0 aggregiert wird, wird diese Dimension nicht in Diagramme eingeschlossen.*

Beispiel	Ergebnis
`MissingCount ([OrderNumber])/MissingCount ({1} Total [OrderNumber])`	Die Formel liefert die Anzahl der unvollständigen Bestellungen des ausgewählten Kunden als Bruchteil der unvollständigen Bestellungen aller Kunden. Insgesamt fehlen für alle Kunden 3 Werte für OrderNumber. Deshalb lautet das Ergebnis für jeden Customer, bei dem ein Wert für Product fehlt, 1/3.

Im Beispiel verwendete Daten:

```
Temp:
LOAD * inline [
Customer|Product|OrderNumber|UnitSales|UnitPrice
Astrida|AA|1|4|16
Astrida|AA|7|10|15
Astrida|BB|4|9|9
Betacab|CC|6|5|10
Betacab|AA|5|2|20
Betacab|BB||| 25
Canutility|AA|||15
Canutility|CC| ||19
Divadip|CC|2|4|16
Divadip|DD|3|1|25
] (delimiter is '|');
```

NullCount

NullCount() liefert die Anzahl der in der Formel aggregierten NULL-Werte, wie in einer Bedingung **group by** definiert wurde.

Syntax:

```
NullCount ( [ distinct ] expr)
```

Rückgabe Datentyp: ganze Zahl

Argumente:

Argumente

Argument	Beschreibung
expr Expression	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.
distinct	Ist der Formel das Wort **distinct** vorangestellt, werden Dubletten nicht berücksichtigt.

Beispiele und Ergebnisse:

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Fügen Sie einem Arbeitsblatt in unserer App dann mindestens diejenigen Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um das Ergebnis anzuzeigen.

Damit die Anzeige genauso wie in der unteren Ergebnisspalte aussieht, schalten Sie im Eigenschaftsfenster unter "Sortierung" von "Auto" auf "Benutzerdefiniert" um und heben Sie anschließend die numerische und alphabetische Sortierung auf.

Skriptbeispiele

Beispiel	Ergebnis

`Set NULLINTERPRET = NULL;`
`Temp:`
`LOAD * inline [`
`Customer|Product|OrderNumber|UnitSales|CustomerID`
`Astrida|AA|1|10|1`
`Astrida|AA|7|18|1`
`Astrida|BB|4|9|1`
`Astrida|CC|6|2|1`
`Betacab|AA|5|4|2`
`Betacab|BB|2|5|2`
`Betacab|DD||`
`Canutility|AA|3|8|`
`Canutility|CC|NULL||`
`] (delimiter is '|');`
`Set NULLINTERPRET=;`
`NullCount1:`
 	
`LOAD Customer,NullCount(OrderNumber) as NullOrdersByCustomer Resident Temp Group By Customer;`
 	
`LOAD NullCount(OrderNumber) as TotalNullCount Resident Temp;`
	`Customer NullOrdersByCustomer`
	
	`Astrida 0`
	
	`Betacab 0`
	
	`Canutility 1`
	
	Der zweite Befehl ergibt:
	`TotalNullCount`
	
	`1`
	in einer Tabelle mit dieser Dimension, da nur ein Datensatz einen NULL-Wert enthält.

NullCount - Diagrammfunktion

NullCount() aggregiert die Anzahl der NULL-Werte nach den Dimensionen des Diagramms.

Syntax:

```
NullCount({[SetExpression][DISTINCT] [TOTAL [<fld {,fld}>]]} expr)
```

Rückgabe Datentyp: ganze Zahl

Argumente:

Argumente

Argument	Beschreibung
expr	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.
set_ expression	Standardmäßig berechnet sich die Aggregierungsfunktion über alle wählbaren Werte. Alternativ können Sie die der Berechnung zugrunde liegenden Werte über die Auswahlformel bestimmen.

Argument	Beschreibung
DISTINCT	Der Zusatz **DISTINCT** vor den Funktionsargumenten bewirkt, dass bei der Auswertung der Funktionsargumente entstehende Duplikate nicht berücksichtigt werden.
TOTAL	Der Zusatz **TOTAL** vor der Funktion bewirkt, dass die Berechnung über alle ausgewählten bzw. wählbaren Werte erfolgt, und nicht nur über diejenigen, die zu dem Wert der aktuellen Dimension zählen. Die Dimensionen des Diagramms werden also nicht berücksichtigt. Mit **TOTAL [<fld {.fld}>]**, wobei auf den Zusatz **TOTAL** eine Liste aus mindestens einem Feldnamen (d. h. einer Teilmenge der Diagrammdimensionsvariablen) folgt, erstellen Sie eine Teilmenge aller möglichen Werte.

Beispiele und Ergebnisse:

Beispiele und Ergebnisse

Beispiel	Ergebnis
NullCount ([OrderNumber])	1, da wir einen NULL-Wert mithilfe von NullInterpret im Inline-**LOAD**-Befehl eingeführt haben.

Im Beispiel verwendete Daten:

```
Set NULLINTERPRET = NULL;
Temp:
LOAD * inline [
Customer|Product|OrderNumber|UnitSales|CustomerID
Astrida|AA|1|10|1
Astrida|AA|7|18|1
Astrida|BB|4|9|1
Astrida|CC|6|2|1
Betacab|AA|5|4|2
Betacab|BB|2|5|2
Betacab|DD|||
Canutility|AA|3|8|
Canutility|CC|NULL||
] (delimiter is '|');
Set NULLINTERPRET=;
```

NumericCount

NumericCount() liefert die Anzahl der in der Formel gefundenen numerischen Werte, wie in einer Bedingung **group by** definiert wurde.

Syntax:

```
NumericCount ( [ distinct ] expr)
```

Rückgabe Datentyp: ganze Zahl

Argumente:

Argumente

Argument	Beschreibung
expr Expression	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.
distinct	Ist der Formel das Wort **distinct** vorangestellt, werden Dubletten nicht berücksichtigt.

Beispiele und Ergebnisse:

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Fügen Sie einem Arbeitsblatt in unserer App dann mindestens diejenigen Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um das Ergebnis anzuzeigen.

Damit die Anzeige genauso wie in der unteren Ergebnisspalte aussieht, schalten Sie im Eigenschaftsfenster unter "Sortierung" von "Auto" auf "Benutzerdefiniert" um und heben Sie anschließend die numerische und alphabetische Sortierung auf.

Skriptbeispiel

Beispiel	Ergebnis
`LOAD NumericCount(OrderNumber) as TotalNumericCount Resident Temp;`	Der zweite Befehl ergibt: TotalNumericCount 7 in einer Tabelle mit dieser Dimension.
Vorgabe: Die Tabelle **Temp** wird wie im vorherigen Beispiel geladen: `LOAD NumericCount(distinct OrderNumber) as TotalNumericCountDistinct Resident Temp;`	TotalNumericCountDistinct 6 Denn es gibt einen OrderNumber, der einen anderen dupliziert. Das Ergebnis sind daher 6 Werte, die keine Duplikate sind.

Beispiel:

```
Temp:

LOAD * inline [

Customer|Product|OrderNumber|UnitSales|UnitPrice

Astrida|AA|1|4|16

Astrida|AA|7|10|15

Astrida|BB|4|9|9

Betacab|CC|6|5|10
```

```
Betacab|AA|5|2|20

Betacab|BB||| 25

Canutility|AA|||15

Canutility|CC| ||19

Divadip|CC|2|4|16

Divadip|DD|7|1|25

] (delimiter is '|');

NumCount1:

LOAD Customer,NumericCount(OrderNumber) as NumericCountByCustomer Resident Temp Group By
Customer;
```

Ergebnistabelle

Customer	NumericCountByCustomer
Astrida	3
Betacab	2
Canutility	0
Divadip	2

NumericCount - Diagrammfunktion

NumericCount() aggregiert die Anzahl der numerischen Werte nach den Dimensionen des Diagramms.

Syntax:

```
NumericCount({[SetExpression] [DISTINCT] [TOTAL [<fld {,fld}>]]} expr)
```

Rückgabe Datentyp: ganze Zahl

Argumente:

Argumente

Argument	Beschreibung
expr	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.
set_ expression	Standardmäßig berechnet sich die Aggregierungsfunktion über alle wählbaren Werte. Alternativ können Sie die der Berechnung zugrunde liegenden Werte über die Auswahlformel bestimmen.

Argument	Beschreibung
DISTINCT	Der Zusatz **DISTINCT** vor den Funktionsargumenten bewirkt, dass bei der Auswertung der Funktionsargumente entstehende Duplikate nicht berücksichtigt werden.
TOTAL	Der Zusatz **TOTAL** vor der Funktion bewirkt, dass die Berechnung über alle ausgewählten bzw. wählbaren Werte erfolgt, und nicht nur über diejenigen, die zu dem Wert der aktuellen Dimension zählen. Die Dimensionen des Diagramms werden also nicht berücksichtigt. Mit **TOTAL [<fld {.fld}>]**, wobei auf den Zusatz **TOTAL** eine Liste aus mindestens einem Feldnamen (d. h. einer Teilmenge der Diagrammdimensionsvariablen) folgt, erstellen Sie eine Teilmenge aller möglichen Werte.

Beispiele und Ergebnisse:

Data

Customer	Product	OrderNumber	UnitSales	Unit Price
Astrida	AA	1	4	16
Astrida	AA	7	10	15
Astrida	BB	4	9	1
Betacab	BB	6	5	10
Betacab	CC	5	2	20
Betacab	DD			25
Canutility	AA			15
Canutility	CC			19
Divadip	AA	2	4	16
Divadip	DD	3		25

Für die folgenden Beispiele wird vorausgesetzt, dass alle Kunden ausgewählt sind, sofern nicht anders angegeben.

Beispiele und Ergebnisse

Beispiel	Ergebnis
`NumericCount ([OrderNumber])`	7, da drei der 10 Felder in OrderNumber leer sind. *"0" zählt als Wert und nicht als leere Zelle. Wenn jedoch eine Kennzahl für eine Dimension auf 0 aggregiert wird, wird diese Dimension nicht in Diagramme eingeschlossen.*
`NumericCount ([Product])`	0, da alle Produktnamen als Text angegeben sind. Normalerweise können Sie hiermit prüfen, ob Textfelder numerische Inhalte aufweisen.
`NumericCount (DISTINCT [OrderNumber])/Count (DISTINCT [OrderNumber)]`	Ermittelt die Anzahl aller distinkten numerischen Bestellnummern und teilt sie durch die Anzahl der numerischen und nicht numerischen Bestellnummern. Dies ergibt 1, wenn alle Feldwerte numerisch sind. Normalerweise können Sie hiermit prüfen, ob alle Feldwerte numerisch sind. In diesem Beispiel gibt es 7 distinkte numerische Werte für OrderNumber von 8 distinkten numerischen und nicht numerischen, wodurch die Formel 0,875 zurückgibt.

Im Beispiel verwendete Daten:

```
Temp:
LOAD * inline [
Customer|Product|OrderNumber|UnitSales|UnitPrice
Astrida|AA|1|4|16
Astrida|AA|7|10|15
Astrida|BB|4|9|9
Betacab|CC|6|5|10
Betacab|AA|5|2|20
Betacab|BB||| 25
Canutility|AA|||15
Canutility|CC| ||19
Divadip|CC|2|4|16
Divadip|DD|3|1|25
] (delimiter is '|');
```

TextCount

TextCount() liefert die Anzahl der Feldwerte, die in der Formel nicht numerisch aggregiert wurden, wie in einer **group by**-Bedingung definiert wurde.

Syntax:

```
TextCount ( [ distinct ] expr)
```

Rückgabe Datentyp: ganze Zahl

Argumente:

Argumente

Argument	Beschreibung
expr Expression	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.
distinct	Ist der Formel das Wort **distinct** vorangestellt, werden Dubletten nicht berücksichtigt.

Beispiele und Ergebnisse:

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Fügen Sie einem Arbeitsblatt in unserer App dann mindestens diejenigen Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um das Ergebnis anzuzeigen.

Damit die Anzeige genauso wie in der unteren Ergebnisspalte aussieht, schalten Sie im Eigenschaftsfenster unter "Sortierung" von "Auto" auf "Benutzerdefiniert" um und heben Sie anschließend die numerische und alphabetische Sortierung auf.

Beispiel:

```
Temp:
LOAD * inline [
Customer|Product|OrderNumber|UnitSales|UnitPrice
Astrida|AA|1|4|16
Astrida|AA|7|10|15
Astrida|BB|4|9|9
Betacab|CC|6|5|10
Betacab|AA|5|2|20
Betacab|BB||| 25
Canutility|AA|||15
Canutility|CC| ||19
Divadip|CC|2|4|16
Divadip|DD|3|1|25
] (delimiter is '|');
TextCount1:
LOAD Customer,TextCount(Product) as ProductTextCount Resident Temp Group By Customer;
```

Ergebnistabelle

Customer	ProductTextCount
Astrida	3
Betacab	3
Canutility	2
Divadip	2

Beispiel:

```
LOAD Customer,TextCount(OrderNumber) as OrderNumberTextCount Resident Temp Group By Customer;
```

Ergebnistabelle

Customer	OrderNumberTextCount
Astrida	0
Betacab	1
Canutility	2
Divadip	0

TextCount - Diagrammfunktion

TextCount() aggregiert die Anzahl der nicht numerischen Feldwerte nach den Dimensionen des Diagramms.

Syntax:

```
TextCount({[SetExpression] [DISTINCT] [TOTAL [<fld {,fld}>]]} expr)
```

Rückgabe Datentyp: ganze Zahl

Argumente:

Argumente

Argument	Beschreibung
expr	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.
SetExpression	Standardmäßig berechnet sich die Aggregierungsfunktion über alle wählbaren Werte. Alternativ können Sie die der Berechnung zugrunde liegenden Werte über die Auswahlformel bestimmen.
DISTINCT	Der Zusatz **DISTINCT** vor den Funktionsargumenten bewirkt, dass bei der Auswertung der Funktionsargumente entstehende Duplikate nicht berücksichtigt werden.
TOTAL	Der Zusatz **TOTAL** vor der Funktion bewirkt, dass die Berechnung über alle ausgewählten bzw. wählbaren Werte erfolgt, und nicht nur über diejenigen, die zu dem Wert der aktuellen Dimension zählen. Die Dimensionen des Diagramms werden also nicht berücksichtigt. Mit **TOTAL [<fld {.fld}>]**, wobei auf den Zusatz **TOTAL** eine Liste aus mindestens einem Feldnamen (d. h. einer Teilmenge der Diagrammdimensionsvariablen) folgt, erstellen Sie eine Teilmenge aller möglichen Werte.

Beispiele und Ergebnisse:

Data

Customer	Product	OrderNumber	UnitSales	Unit Price
Astrida	AA	1	4	16
Astrida	AA	7	10	15
Astrida	BB	4	9	1
Betacab	BB	6	5	10
Betacab	CC	5	2	20
Betacab	DD			25
Canutility	AA			15
Canutility	CC			19
Divadip	AA	2	4	16
Divadip	DD	3		25

Beispiele und Ergebnisse

Beispiel	Ergebnis
`TextCount ([Product])`	10, da alle 10 Felder in Product Text aufweisen. *"0" zählt als Wert und nicht als leere Zelle. Wenn jedoch eine Kennzahl für eine Dimension auf 0 aggregiert wird, wird diese Dimension nicht in Diagramme eingeschlossen. Leere Zellen werden nicht als Textfelder gewertet und somit nicht von TextCount gezählt.*
`TextCount ([OrderNumber])`	3, da leere Zellen gezählt werden. Normalerweise können Sie hiermit prüfen, ob numerische Felder Textwerte beinhalten oder nicht Null sind.
`TextCount (DISTINCT [Product])/Count ([Product)]`	Ermittelt die Zahl der distinkten Textwerte von Product (4) und teilt das Ergebnis durch die Gesamtanzahl der Werte in Product (10). Das Ergebnis ist 0,4.

Im Beispiel verwendete Daten:

```
Temp:
LOAD * inline [
Customer|Product|OrderNumber|UnitSales|UnitPrice
Astrida|AA|1|4|16
Astrida|AA|7|1|15
Astrida|BB|4|9|9
```

```
Betacab|CC|6|5|10
Betacab|AA|5|2|20
Betacab|BB|||| 25
Canutility|AA|||15
Canutility|CC|||19
Divadip|CC|2|4|16
Divadip|DD|3|1|25
] (delimiter is '|');
```

Finanz-Aggregierung

In diesem Abschnitt werden Aggregierungsfunktionen in Hinblick auf finanzielle Themen wie Zahlungen und Cashflow beschreiben.

Jede Funktion wird nach der Übersicht genauer beschrieben. Sie können auch auf den Funktionsnamen in der Syntax klicken, um direkt auf die Details zu der spezifischen Funktion zuzugreifen.

Finanz-Aggregierungsfunktionen im Datenladeskript verwenden

IRR

IRR() liefert den internen Zinsfuß einer Investition für eine Reihe von Zahlungen, repräsentiert durch die Werte der Formel, über die im group by-Zusatz bezeichneten Datensätze.

```
IRR (expression)
```

XIRR

Die Funktion **XIRR()** gibt den aggregierten (jährlichen) internen Zinsfuß einer Investition für eine Reihe geplanter (nicht unbedingt regelmäßig erfolgender) Geldflüsse zurück, repräsentiert durch die Wertepaare aus **pmt** und **date** und über die im Zusatz group by bezeichneten Datensätze iteriert. Alle Beträge werden auf ein 365-Tage-Jahr hochgerechnet.

```
XIRR (valueexpression, dateexpression )
```

NPV

Die Skriptfunktion **NPV()** nimmt einen Diskontsatz und mehrere Werte, sortiert nach Zeitraum. Eingänge (Einnahmen) sind positive Werte, Ausgänge (zukünftige Zahlungen) sind negative Werte für diese Berechnungen. Sie finden am Ende jedes Zeitraums statt.

```
NPV (rate, expression)
```

XNPV

Die Funktion **XNPV()** gibt den aggregierten Nettobarwert einer Reihe geplanter (nicht unbedingt regelmäßig erfolgender) Geldflüsse zurück, repräsentiert durch die Wertepaare in **pmt** und **date**. Alle Beträge werden auf ein 365-Tage-Jahr hochgerechnet.

```
XNPV (rate, valueexpression, dateexpression)
```

Finanz-Aggregierungsfunktionen in Diagrammformeln verwenden

Diese Finanz-Aggregierungsfunktionen können in Diagrammen verwendet werden.

IRR

IRR() liefert den internen Zinsfuß, Internal Rate of Return, einer Investition für eine Reihe von Zahlungen, repräsentiert durch **value** der Formel und über die Dimensionen des Diagramms.

```
IRR - Diagrammfunktion[TOTAL [<fld {,fld}>]] value)
```

NPV

NPV() liefert den Kapitalwert einer Investition, Net Present Value, basierend auf dem **discount_rate** pro Zeitraum und einer Reihe regelmäßig erfolgender Einzahlungen (positive Werte) und Auszahlungen (negative Werte), repräsentiert durch die Werte von **value**, aggregiert nach den Dimensionen des Diagramms. Es wird angenommen, dass eingehende und ausgehende Zahlungen am Ende des jeweiligen Zeitraums stattfinden.

```
NPV - Diagrammfunktion([TOTAL [<fld {,fld}>]] discount_rate, value)
```

XIRR

XIRR() gibt den aggregierten internen (jährlichen) Zinsfuß einer Investition für eine Reihe geplanter (nicht unbedingt regelmäßig erfolgender) Geldflüsse zurück, repräsentiert durch Wertepaare in den Formeln aus **pmt** und **date** und über die Diagrammdimensionen iteriert. Alle Beträge werden auf ein 365-Tage-Jahr hochgerechnet.

```
XIRR - Diagrammfunktion([TOTAL [<fld {,fld}>]] pmt, date)
```

XNPV

XNPV() liefert den aggregierten Nettobarwert einer Reihe (nicht notwendigerweise regelmäßig erfolgenden) Cashflows, repräsentiert durch die Wertepaare in den Formeln von **pmt** und **date**, iteriert nach den Dimensionen des Diagramms. Alle Beträge werden auf ein 365-Tage-Jahr hochgerechnet.

```
XNPV - Diagrammfunktion([TOTAL [<fld{,fld}>]] discount_rate, pmt, date)
```

IRR

IRR() liefert den internen Zinsfuß einer Investition für eine Reihe von Zahlungen, repräsentiert durch die Werte der Formel, über die im group by-Zusatz bezeichneten Datensätze.

Diese Zahlungsströme müssen nicht gleich hoch sein, so wie sie es im Falle einer Annuität wären. Sie müssen jedoch in regelmäßigen Intervallen erfolgen, beispielsweise monatlich oder jährlich. Der interne Zinsfuß ist der Zinssatz einer Investition bei regelmäßig erfolgenden Auszahlungen (negative Werte) und Einzahlungen (positive Werte). Die Funktion benötigt mindestens einen positiven und einen negativen Wert, um ein Ergebnis berechnen zu können.

Diese Funktion verwendet eine vereinfachte Version der Newton-Methode zum Berechnen des internen Zinsfußes (IRR).

Syntax:

```
IRR(value)
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
value	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.

Beschränkungen:

Textwerte, NULL-Werte und fehlende Werte werden ignoriert.

Beispiele und Ergebnisse:

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Fügen Sie einem Arbeitsblatt in Ihrer App dann die Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um das Ergebnis anzuzeigen.

Beispiele und Ergebnisse:

Beispiele und Ergebnisse

Beispiel	Jahr	IRR2013

`Cashflow:`
`LOAD 2013 as Year, * inline [`
`Date|Discount|Payments`
`2013-01-01|0.1|-10000`
`2013-03-01|0.1|3000`
`2013-10-30|0.1|4200`
`2014-02-01|0.2|6800`
`] (delimiter is '|');`
 	
`Cashflow1:`
`LOAD Year,IRR(Payments) as IRR2013 Resident Cashflow Group By Year;`
	2013	0.1634

IRR - Diagrammfunktion

IRR() liefert den internen Zinsfuß, Internal Rate of Return, einer Investition für eine Reihe von Zahlungen, repräsentiert durch **value** der Formel und über die Dimensionen des Diagramms.

Diese Zahlungsströme müssen nicht gleich hoch sein, so wie sie es im Falle einer Annuität wären. Sie müssen jedoch in regelmäßigen Intervallen erfolgen, beispielsweise monatlich oder jährlich. Der interne Zinsfuß ist der Zinssatz einer Investition bei regelmäßig erfolgenden Auszahlungen (negative Werte) und Einzahlungen (positive Werte). Die Funktion benötigt mindestens einen positiven und einen negativen Wert, um ein Ergebnis berechnen zu können.

Diese Funktion verwendet eine vereinfachte Version der Newton-Methode zum Berechnen des internen Zinsfußes (IRR).

Syntax:

```
IRR([TOTAL [<fld {,fld}>]] value)
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
value	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.
TOTAL	Der Zusatz **TOTAL** vor der Funktion bewirkt, dass die Berechnung über alle ausgewählten bzw. wählbaren Werte erfolgt, und nicht nur über diejenigen, die zu dem Wert der aktuellen Dimension zählen. Die Dimensionen des Diagramms werden also nicht berücksichtigt. Mit **TOTAL [<fld {.fld}>]**, wobei auf den Zusatz **TOTAL** eine Liste aus mindestens einem Feldnamen (d. h. einer Teilmenge der Diagrammdimensionsvariablen) folgt, erstellen Sie eine Teilmenge aller möglichen Werte.

Beschränkungen:

Die Parameter der Aggregierungsfunktion dürfen keine anderen Aggregierungsfunktionen enthalten, es sei denn, diese inneren Aggregierungen enthalten den Qualifizierer **TOTAL**. Für komplexere verschachtelte Aggregierungen verwenden Sie die erweiterte Funktion **Aggr** in Verbindung mit einer angegebenen Dimension.

Textwerte, NULL-Werte und fehlende Werte werden ignoriert.

Beispiele und Ergebnisse:

Beispiele und Ergebnisse

Beispiel	Ergebnis
`IRR (Payments)`	0.1634 Es wird angenommen, dass die Zahlungen regelmäßig, z. B. monatlich, erfolgen. *Das Feld für Datum wird im XIRR-Beispiel verwendet, in dem Zahlungen auch unregelmäßig erfolgen können, wenn Sie das jeweilige Datum der Zahlung angeben.*

In Beispielen verwendete Daten:

```
Cashflow:
LOAD 2013 as Year, * inline [
Date|Discount|Payments
2013-01-01|0.1|-10000
2013-03-01|0.1|3000
2013-10-30|0.1|4200
2014-02-01|0.2|6800
] (delimiter is '|');
```

Siehe auch:

XIRR - Diagrammfunktion (page 399)
Aggr - Diagrammfunktion (page 566)

NPV

Die Skriptfunktion **NPV()** nimmt einen Diskontsatz und mehrere Werte, sortiert nach Zeitraum. Eingänge (Einnahmen) sind positive Werte, Ausgänge (zukünftige Zahlungen) sind negative Werte für diese Berechnungen. Sie finden am Ende jedes Zeitraums statt.

Der Nettobarwert (Net Present Value, NPV) dient zum Berechnen des aktuellen Gesamtwerts von zukünftigen Geldflüssen. Zur Berechnung des NPV müssen die zukünftigen Geldflüsse für jeden Zeitraum geschätzt und die korrekte Diskontierungsrate bestimmt werden. Die Skriptfunktion **NPV()** enthält eine Diskontierungsrate und mehrere Werte, sortiert nach Zeitraum. Eingehende Geldflüsse (Einnahmen) sind positive Werte, ausgehende Geldflüsse (zukünftige Zahlungen) sind negative Werte für diese Berechnungen. Sie finden am Ende jedes Zeitraums statt.

Syntax:

```
NPV(discount_rate, value)
```

Rückgabe Datentyp: numerisch. Standardmäßig wird das Ergebnis als Währung formatiert.

Der Nettobarwert wird mit folgender Formel berechnet:

$$NPV = \sum_{t=1}^{n} \frac{R_t}{(1+i)^t}$$

Dabei gilt:

- R_t = eingehende und ausgehende Nettogeldflüsse während eines einzelnen Zeitraums `t`
- `i` = Diskontierungsrate oder Rendite, die in alternativen Anlagen verdient werden könnte
- `t` = Anzahl der Zeiträume

Argumente

Argument	Beschreibung
discount_ rate	**discount_rate** ist der Prozentsatz des angewendeten Rabatts. Ein Wert von 0,1 gibt eine Diskontierungsrate von 10 % an.
value	Dieses Feld enthält Werte für mehrere Zeiträume, nach Zeitraum sortiert. Der erste Wert steht für den Geldfluss am Ende von Zeitraum 1, usw.

Beschränkungen:

Die Funktion `NPV()` weist die folgenden Beschränkungen auf:

- Textwerte, NULL-Werte und fehlende Werte werden ignoriert.
- Geldflusswerte müssen in aufsteigender Zeitraumreihenfolge sortiert sein.

Verwendung

NPV() ist einen Finanzfunktion und dient zum Prüfen der Projektrentabilität und zum Ableiten anderer Kennzahlen. Die Funktion ist nützlich, wenn Geldflüsse als Rohdaten vorliegen.

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung SET DateFormat in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Beispiel 1 – Einmalige Zahlung (Skript)

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz für ein Projekt und dessen Geldfluss für einen Zeitraum wird in eine Tabelle namens CashFlow geladen.
- Ein Resident Load aus der Tabelle CashFlow wird verwendet, um das NPV-Feld für das Projekt in der Tabelle NPV zu berechnen.
- Eine feste Diskontierungsrate von 10 % wird in der NPV-Berechnung verwendet.
- Ein Group By-Befehl wird zum Gruppieren aller Zahlungen für das Projekt verwendet.

Ladeskript

```
CashFlow:
Load
*
Inline
[
PrjId,PeriodId,Values
1,1,1000
];
```

```
NPV:
Load
       PrjId,
       NPV(0.1,Values) as NPV //Discount Rate of 10%
Resident CashFlow
Group By PrjId;
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `PrjId`
- `NPV`

Ergebnistabelle

PrjId	NPV
1	$909.09

Für eine einmalige Zahlung von $1000, die am Ende eines Zeitraums erhalten werden soll, und mit einer Diskontierungsrate von 10 % pro Zeitraum entspricht der NPV $1000 geteilt durch (1 + Diskontierungsrate). Der effektive NPV entspricht $909,09

Beispiel 2 – Mehrere Zahlungen (Skript)

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz für ein Projekt und dessen Geldflüsse für mehrere Zeiträume wird in eine Tabelle namens `CashFlow` geladen.
- Ein Resident Load aus der Tabelle `CashFlow` wird verwendet, um das NPV-Feld für das Projekt in der Tabelle `NPV` zu berechnen.
- Eine feste Diskontierungsrate von 10 % (0,1) wird in der NPV-Berechnung verwendet.
- Ein `Group By`-Befehl wird zum Gruppieren aller Zahlungen für das Projekt verwendet.

Ladeskript

```
CashFlow:
Load
*
Inline
[
PrjId,PeriodId,Values
```

```
1,1,1000
1,2,1000
];

NPV:
Load
       PrjId,
       NPV(0.1,Values) as NPV //Discount Rate of 10%
Resident CashFlow
Group By PrjId;
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `PrjId`
- `NPV`

Ergebnistabelle

PrjId	NPV
1	$1735.54

Für Zahlungen von $1000, die am Ende von zwei Zeiträumen erhalten werden sollen, und mit einer Diskontierungsrate von 10 % pro Zeitraum entspricht der NPV $1735,54.

Beispiel 3 – Mehrere Zahlungen (Skript)

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Diskontierungsraten für zwei Projekte werden in eine Tabelle namens `Project` geladen.
- Cashflows für mehrere Zeiträume für jedes Projekt nach Projekt-ID und Zeitraum-ID. Diese Zeitraum-ID kann zum Sortieren der Datensätze verwendet werden, wenn die Daten nicht sortiert sind.
- Die Kombination aus `NoConcatenate`, Resident Loads und der Funktion `Left Join` dient zum Erstellen einer temporären Tabelle, `tmpNPV`. Die Tabelle kombiniert die Datensätze der Tabellen `Project` und `CashFlow` in einer flachen Tabelle. In dieser Tabelle werden Diskontierungsraten für jeden Zeitraum wiederholt.
- Ein Resident Load aus der Tabelle `tmpNPV` wird verwendet, um das NPV-Feld für jedes Projekt in der Tabelle `NPV` zu berechnen.
- Die Diskontierungsrate mit einem Wert ist jedem Projekt zugeordnet. Sie wird mit der Funktion `only()`

abgerufen und in der NPV-Berechnung für jedes Projekt verwendet.

- Ein `Group By`-Befehl wird zum Gruppieren aller Zahlungen für jedes Projekt nach Projekt-ID verwendet.

Um zu verhindern, dass synthetische oder redundante Daten in das Datenmodell geladen werden, wird die Tabelle `tmpNPV` am Ende des Skripts verworfen.

Ladeskript

```
Project:
Load * inline [
PrjId,Discount_Rate
1,0.1
2,0.15
];

CashFlow:
Load
*
Inline
[
PrjId,PeriodId,Values
1,1,1000
1,2,1000
1,3,1000
2,1,500
2,2,500
2,3,1000
2,4,1000
];

tmpNPV:
NoConcatenate Load *
Resident Project;
Left Join
Load *
Resident CashFlow;

NPV:
Load
       PrjId,
       NPV(Only(Discount_Rate),Values) as NPV //Discount Rate will be 10% for Project 1 and 15% for
Project 2
Resident tmpNPV
Group By PrjId;

Drop table tmpNPV;
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- PrjId
- NPV

Ergebnistabelle

PrjId	NPV
1	$2486.85
2	$2042.12

Für Projekt-ID 1 werden Zahlungen von $1000 am Ende von drei Zeiträumen mit einer Diskontierungsrate von 10 % pro Zeitraum erwartet. Daher entspricht der effektive NPV $2486,85.

Für Projekt-ID 2 werden zwei Zahlungen von $500 und zwei weitere Zahlungen von $1000 über vier Zeiträume hinweg mit einer Diskontierungsrate von 15 % erwartet. Daher entspricht der effektive NPV $2042,12.

Beispiel 4 – Beispiel für Projektrentabilität (Skript)

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Diskontierungsraten und anfängliche Investitionen (Zeitraum 0) für zwei Projekte werden in eine Tabelle namens `Project` geladen.
- Cashflows für mehrere Zeiträume für jedes Projekt nach Projekt-ID und Zeitraum-ID. Diese Zeitraum-ID kann zum Sortieren der Datensätze verwendet werden, wenn die Daten nicht sortiert sind.
- Die Kombination aus `NoConcatenate`, Resident Loads und der Funktion `Left Join` dient zum Erstellen einer temporären Tabelle, `tmpNPV`. Die Tabelle kombiniert die Datensätze der Tabellen `Project` und `CashFlow` in einer flachen Tabelle. In dieser Tabelle werden Diskontierungsraten für jeden Zeitraum wiederholt.
- Die Diskontierungsrate mit einem Wert, die jedem Projekt zugewiesen ist, wird anhand der Funktion `only()` abgerufen und in der NPV-Berechnung für jedes Projekt verwendet.
- Ein Resident Load aus der Tabelle tmpNPV wird verwendet, um das NPV-Feld für jedes Projekt in einer Tabelle NPV zu berechnen.
- Ein weiteres Feld, das den NPV durch die anfängliche Investition für jedes Projekt teilt, wird erstellt, um den Projektrentabilitätsindex zu berechnen.
- Ein Group By-Befehl gruppiert nach Projekt-ID und wird zum Gruppieren aller Zahlungen für jedes Projekt verwendet.

Um zu verhindern, dass synthetische oder redundante Daten in das Datenmodell geladen werden, wird die Tabelle `tmpNPV` am Ende des Skripts verworfen.

Ladeskript

```
Project:
Load * inline [
PrjId,Discount_Rate, Initial_Investment
1,0.1,100000
2,0.15,100000
];

CashFlow:
Load
*
Inline
[
PrjId,PeriodId,Values,
1,1,35000
1,2,35000
1,3,35000
2,1,30000
2,2,40000
2,3,50000
2,4,60000
];

tmpNPV:
NoConcatenate Load *
Resident Project;
Left Join
Load *
Resident CashFlow;

NPV:
Load
        PrjId,
        NPV(Only(Discount_Rate),Values) as NPV, //Discount Rate will be 10% for Project 1 and
15% for Project 2
    NPV(Only(Discount_Rate),Values)/ Only(Initial_Investment) as Profitability_Index
Resident tmpNPV
Group By PrjId;

Drop table tmpNPV;
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- PrjId
- NPV

Erstellen Sie die folgende Kennzahl:

```
=only(Profitability_Index)
```

Ergebnistabelle

PrjId	NPV	=only(Profitability_Index)
1	$87039.82	0.87
2	$123513.71	1.24

Projekt-ID 1 hat einen effektiven NPV von $87039.82 und eine anfängliche Investition von $100000. Daher entspricht der Rentabilitätsindex 0,87. Da dieser Wert kleiner als 1 ist, ist das Projekt nicht rentabel.

Projekt-ID 2 hat einen effektiven NPV von $123513,71 und eine anfängliche Investition von $100000. Daher entspricht der Rentabilitätsindex 1,24. Da dieser Wert größer als 1 ist, ist das Projekt rentabel.

NPV - Diagrammfunktion

NPV() liefert den Kapitalwert einer Investition, Net Present Value, basierend auf dem **discount_rate** pro Zeitraum und einer Reihe regelmäßig erfolgender Einzahlungen (positive Werte) und Auszahlungen (negative Werte), repräsentiert durch die Werte von **value**, aggregiert nach den Dimensionen des Diagramms. Es wird angenommen, dass eingehende und ausgehende Zahlungen am Ende des jeweiligen Zeitraums stattfinden.

Syntax:

```
NPV([TOTAL [<fld {,fld}>]] discount_rate, value)
```

Rückgabe Datentyp: numerisch Standardmäßig wird das Ergebnis als Währung formatiert.

Argumente:

Argumente

Argument	Beschreibung
discount_rate	**discount_rate** ist der Prozentsatz des angewendeten Rabatts.
value	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.
TOTAL	Der Zusatz **TOTAL** vor der Funktion bewirkt, dass die Berechnung über alle ausgewählten bzw. wählbaren Werte erfolgt, und nicht nur über diejenigen, die zu dem Wert der aktuellen Dimension zählen. Die Dimensionen des Diagramms werden also nicht berücksichtigt. Mit **TOTAL [<fld {.fld}>]**, wobei auf den Zusatz **TOTAL** eine Liste aus mindestens einem Feldnamen (d. h. einer Teilmenge der Diagrammdimensionsvariablen) folgt, erstellen Sie eine Teilmenge aller möglichen Werte. Auf den Zusatz **TOTAL** kann eine Reihe von Feldnamen in spitzen Klammern folgen. Diese Feldnamen sollten eine Teilmenge der Dimensionen des Diagramms sein. der Funktion nur noch die explizit aufgeführten Dimensionen berücksichtigt, d. h. für jede Kombination von Werten dieser Felder wird ein Formelwert berechnet. Es können auch Felder aufgeführt werden, die nicht Dimension des Diagramms sind. Dies ist sinnvoll, wenn Gruppen als Dimension dienen. Führt man alle Variablen der Gruppe auf, kommt dies erst beim Wechsel des Drilldown zum Tragen.

Beschränkungen:

discount_rate und **value** dürfen keine Aggregierungsfunktionen ohne den Zusatz **TOTAL** enthalten. Für komplexere verschachtelte Aggregierungen verwenden Sie die erweiterte Funktion **Aggr** in Verbindung mit einer angegebenen Dimension.

Textwerte, NULL-Werte und fehlende Werte werden ignoriert.

Beispiele und Ergebnisse:

Beispiele und Ergebnisse

Beispiel	Ergebnis
`NPV(Discount, Payments)`	-$540.12

In Beispielen verwendete Daten:

```
Cashflow:
LOAD 2013 as Year, * inline [
Date|Discount|Payments
2013-01-01|0.1|-10000
2013-03-01|0.1|3000
2013-10-30|0.1|4200
2014-02-01|0.2|6800
] (delimiter is '|');
```

Siehe auch:

XNPV - Diagrammfunktion (page 408)
Aggr - Diagrammfunktion (page 566)

XIRR

Die Funktion **XIRR()** gibt den aggregierten (jährlichen) internen Zinsfuß einer Investition für eine Reihe geplanter (nicht unbedingt regelmäßig erfolgender) Geldflüsse zurück, repräsentiert durch die Wertepaare aus **pmt** und **date** und über die im Zusatz group by bezeichneten Datensätze iteriert. Alle Beträge werden auf ein 365-Tage-Jahr hochgerechnet.

Die XIRR-Funktionalität von Qlik (die Funktionen **XIRR()** und **RangeXIRR()**) verwendet die folgende Gleichung, die nach dem Wert `Rate` aufgelöst wird, um den korrekten XIRR-Wert zu bestimmen:

```
XNPV(Rate, pmt, date) = 0
```

Die Gleichung wird anhand einer vereinfachten Version der Newton-Methode aufgelöst.

Syntax:

```
XIRR(pmt, date )
```

Rückgabe Datentyp: numerisch

Argumente

Argument	Beschreibung
pmt	Zahlungen. Die Formel oder das Feld mit den Cashflows, die den Zahlungszeitpunkten in **date** entsprechen.
date	Die Formel oder das Feld mit den Zeitpunkten, die den Cashflow-Zahlungen in **pmt** entsprechen.

Bei der Arbeit mit dieser Funktion gelten die folgenden Einschränkungen:

- Textwerte, NULL-Werte und fehlende Werte in einem oder beiden Teilen eines Wertepaars führen dazu, dass das Wertepaar ignoriert wird.
- Diese Funktion erfordert mindestens eine gültige negative und einen gültige positive Zahlung (mit entsprechendem gültigem Datum). Wenn diese Zahlungen nicht angegeben werden, wird ein Wert von NULL zurückgegeben.

Die folgenden Themen können Sie bei der Arbeit mit dieser Funktion unterstützen:

- *XNPV (page 402)*: Verwenden Sie diese Funktion, um den aggregierten Nettobarwert für geplante Geldflüsse zu berechnen.
- *RangeXIRR (page 1398)*: **RangeXIRR()** ist die äquivalente Bereichsfunktion für die Funktion **XIRR()**.

In den einzelnen Versionen von Qlik Sense Client-Managed bestehen Unterschiede bei dem zugrunde liegenden Algorithmus, der von dieser Funktion verwendet wird. Informationen zu kürzlichen Aktualisierungen am Algorithmus finden Sie im Support-Artikel über XIRR-Funktion – Behebung und Aktualisierung*.*

Beispiel

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Transaktionsdaten für eine Reihe von Geldflüssen.
- Verwendung der Funktion **XIRR()** zum Berechnen des jährlichen internen Zinsfußes für diese Geldflüsse.

Ladeskript

```
Cashflow:
LOAD 2013 as Year, * inline [
Date|Payments
2013-01-01|-10000
2013-03-01|3000
2013-10-30|4200
2014-02-01|6800
] (delimiter is '|');

Cashflow1:
LOAD Year,XIRR(Payments, Date) as XIRR2013  Resident Cashflow Group By Year;
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `Year`
- `XIRR2013`

Ergebnistabelle

Jahr	XIRR2013
2013	0.5385

Interpretieren des von XIRR zurückgegebenen Werts

Die XIRR-Funktion wird in der Regel zum Analysieren einer Investition verwendet, die mit einer ausgehenden (negativen) Zahlung beginnt, gefolgt von einer Reihe kleinerer eingehender (positiver) Zahlungen. Das folgende vereinfachte Beispiel weist nur eine negative und eine positive Zahlung auf:

```
Cashflow:
LOAD * inline [
Date|Payments
2023-01-01|-100
2024-01-01|110
] (delimiter is '|');
```

Wir leisten eine anfängliche Zahlung von 100 und erhalten nach genau einem Jahr 110 zurück. Dies stellt einen Zinsfuß von 10 % pro Jahr dar. `XIRR(Payments, Date)` gibt einen Wert von 0.1 zurück.

Der von der XIRR-Funktion zurückgegebene Wert kann positiv oder negativ sein. Im Fall einer Investition weist ein negatives Ergebnis darauf hin, dass die Investition ein Verlust war. Der Gewinn- oder Verlustbetrag kann berechnet werden, indem einfach eine Summenaggregierung für das Zahlungsfeld vorgenommen wird.

Im obigen Beispiel verleihen wir unser Geld für ein Jahr. Der Zinsfuß stellt die Zinseinnahmen dar. Die XIRR-Funktion kann auch verwendet werden, wenn Sie sich auf der anderen Seite der Transaktion befinden – wenn Sie sich also Geld leihen, anstatt es zu verleihen.

Betrachten wir dieses Beispiel:

```
Cashflow:
LOAD * inline [
Date|Payments
2023-01-01|100
2024-01-01|-110
] (delimiter is '|');
```

Es entspricht dem ersten Beispiel, aber umgekehrt. Hier leihen wir uns 100 für ein Jahr und zahlen den Betrag mit 10 % Zinsen zurück. In diesem Beispiel gibt die XIRR-Berechnung 0.1 (10 %) zurück, den gleichen Wert wie im ersten Beispiel.

Beachten Sie, dass wir im ersten Beispiel einen Gewinn von 10 erhalten haben, während wir im zweiten Beispiel einen Verlust von 10 hatten. Aber der von der XIRR-Funktion zurückgegebene Wert ist in beiden Beispielen positiv. Das liegt daran, dass die XIRR-Funktion den verdeckten Zinssatz in der Transaktion berechnet, unabhängig davon, auf welcher Seite der Transaktion Sie stehen.

Einschränkungen bei mehreren Lösungen

Die XIRR-Funktion von Qlik wird durch die folgende Gleichung definiert, in der der Wert `Rate` aufgelöst wird:

```
XNPV(Rate, pmt, date) = 0
```

Manchmal kann diese Gleichung mehr als eine Lösung haben. Dies ist als „Problem mit mehreren internen Zinsfüßen“ bekannt und wird durch einen nicht normalen Geldfluss (auch als unkonventioneller Geldfluss bezeichnet) verursacht. Das folgende Ladeskript zeigt ein Beispiel:

```
Cashflow:
LOAD * inline [
Date|Payments
2021-01-01|-200
2022-01-01|500
2023-01-01|-250
] (delimiter is '|');
```

In diesem Beispiel gibt es eine negative und eine positive Lösung (`Rate` = -0.3 und `Rate` = 0.8). **XIRR()** gibt 0.8 zurück.

Wenn die XIRR-Funktion von Qlik nach einer Lösung sucht, beginnt sie bei `Rate` = 0 und erhöht den Zinsfuß schrittweise, bis sie eine Lösung findet. Wenn es mehr als eine positive Lösung gibt, gibt die Funktion die erste gefundene Lösung zurück. Wenn keine positive Lösung gefunden wird, setzt die Funktion `Rate` auf Null zurück und beginnt, in negativer Richtung nach einer Lösung zu suchen.

Beachten Sie, dass es bei „normalem“ Geldfluss garantiert nur eine Lösung gibt. „Normaler“ Geldfluss bedeutet, dass sich alle Zahlungen mit dem gleichen Vorzeichen (positiv oder negativ) in einer fortlaufenden Gruppe befinden.

Siehe auch:

- *XNPV (page 402)*
- *RangeXIRR (page 1398)*
- XIRR-Funktion – Behebung und Aktualisierung

XIRR - Diagrammfunktion

XIRR() gibt den aggregierten internen (jährlichen) Zinsfuß einer Investition für eine Reihe geplanter (nicht unbedingt regelmäßig erfolgender) Geldflüsse zurück, repräsentiert durch Wertepaare in den Formeln aus **pmt** und **date** und über die Diagrammdimensionen iteriert. Alle Beträge werden auf ein 365-Tage-Jahr hochgerechnet.

Die XIRR-Funktionalität von Qlik (die Funktionen **XIRR()** und **RangeXIRR()**) verwendet die folgende Gleichung, die nach dem Wert `Rate` aufgelöst wird, um den korrekten XIRR-Wert zu bestimmen:

```
XNPV(Rate, pmt, date) = 0
```

Die Gleichung wird anhand einer vereinfachten Version der Newton-Methode aufgelöst.

Syntax:

```
XIRR([TOTAL [<fld {,fld}>]] pmt, date)
```

Rückgabe Datentyp: numerisch

Argumente

Argument	Beschreibung
pmt	Zahlungen. Die Formel oder das Feld mit den Cashflows, die den Zahlungszeitpunkten in **date** entsprechen.
date	Die Formel oder das Feld mit den Zeitpunkten, die den Cashflow-Zahlungen in **pmt** entsprechen.
TOTAL	Der Zusatz **TOTAL** vor der Funktion bewirkt, dass die Berechnung über alle ausgewählten bzw. wählbaren Werte erfolgt, und nicht nur über diejenigen, die zu dem Wert der aktuellen Dimension zählen. Die Dimensionen des Diagramms werden also nicht berücksichtigt. Mit **TOTAL [<fld {.fld}>]**, wobei auf den Zusatz **TOTAL** eine Liste aus mindestens einem Feldnamen (d. h. einer Teilmenge der Diagrammdimensionsvariablen) folgt, erstellen Sie eine Teilmenge aller möglichen Werte.

Bei der Arbeit mit dieser Funktion gelten die folgenden Einschränkungen:

- **pmt** und **date** dürfen keine Aggregierungsfunktionen ohne den Zusatz **TOTAL** enthalten. Für komplexere verschachtelte Aggregierungen verwenden Sie die erweiterte Funktion **Aggr** in Verbindung mit einer angegebenen Dimension.
- Textwerte, NULL-Werte und fehlende Werte in einem oder beiden Teilen eines Wertepaars führen dazu, dass das Wertepaar ignoriert wird.
- Diese Funktion erfordert mindestens eine gültige negative und einen gültige positive Zahlung (mit entsprechendem gültigem Datum). Wenn diese Zahlungen nicht angegeben werden, wird ein Wert von NULL zurückgegeben.

Die folgenden Themen können Sie bei der Arbeit mit dieser Funktion unterstützen:

- *XNPV - Diagrammfunktion (page 408)*: Verwenden Sie diese Funktion, um den aggregierten Nettobarwert für geplante Geldflüsse zu berechnen.
- *RangeXIRR (page 1398)*: **RangeXIRR()** ist die äquivalente Bereichsfunktion für die Funktion **XIRR()**.

> *In den einzelnen Versionen von Qlik Sense Client-Managed bestehen Unterschiede bei dem zugrunde liegenden Algorithmus, der von dieser Funktion verwendet wird. Informationen zu kürzlichen Aktualisierungen am Algorithmus finden Sie im Support-Artikel über* XIRR-Funktion – Behebung und Aktualisierung.

Beispiel

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz enthält Geldflusstransaktionen.
- Die Informationen sind in einer Tabelle mit dem Namen `Cashflow` gespeichert.

Ladeskript

```
Cashflow:
LOAD 2013 as Year, * inline [
Date|Payments
2013-01-01|-10000
2013-03-01|3000
2013-10-30|4200
2014-02-01|6800
] (delimiter is '|');
```

Ergebnisse

Gehen Sie folgendermaßen vor:

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Berechnungen als Kennzahl hinzu:

```
=XIRR(Payments, Date)
```

Ergebnistabelle

=XIRR(Payments, Date)
0.5385

Interpretieren des von XIRR zurückgegebenen Werts

Die XIRR-Funktion wird in der Regel zum Analysieren einer Investition verwendet, die mit einer ausgehenden (negativen) Zahlung beginnt, gefolgt von einer Reihe kleinerer eingehender (positiver) Zahlungen. Das folgende vereinfachte Beispiel weist nur eine negative und eine positive Zahlung auf:

```
Cashflow:
LOAD * inline [
Date|Payments
2023-01-01|-100
2024-01-01|110
] (delimiter is '|');
```

Wir leisten eine anfängliche Zahlung von 100 und erhalten nach genau einem Jahr 110 zurück. Dies stellt einen Zinsfuß von 10 % pro Jahr dar. `XIRR(Payments, Date)` gibt einen Wert von 0.1 zurück.

Der von der XIRR-Funktion zurückgegebene Wert kann positiv oder negativ sein. Im Fall einer Investition weist ein negatives Ergebnis darauf hin, dass die Investition ein Verlust war. Der Gewinn- oder Verlustbetrag kann berechnet werden, indem einfach eine Summenaggregierung für das Zahlungsfeld vorgenommen wird.

Im obigen Beispiel verleihen wir unser Geld für ein Jahr. Der Zinsfuß stellt die Zinseinnahmen dar. Die XIRR-Funktion kann auch verwendet werden, wenn Sie sich auf der anderen Seite der Transaktion befinden – wenn Sie sich also Geld leihen, anstatt es zu verleihen.

Betrachten wir dieses Beispiel:

```
Cashflow:
LOAD * inline [
Date|Payments
2023-01-01|100
2024-01-01|-110
] (delimiter is '|');
```

Es entspricht dem ersten Beispiel, aber umgekehrt. Hier leihen wir uns 100 für ein Jahr und zahlen den Betrag mit 10 % Zinsen zurück. In diesem Beispiel gibt die XIRR-Berechnung 0.1 (10 %) zurück, den gleichen Wert wie im ersten Beispiel.

Beachten Sie, dass wir im ersten Beispiel einen Gewinn von 10 erhalten haben, während wir im zweiten Beispiel einen Verlust von 10 hatten. Aber der von der XIRR-Funktion zurückgegebene Wert ist in beiden Beispielen positiv. Das liegt daran, dass die XIRR-Funktion den verdeckten Zinssatz in der Transaktion berechnet, unabhängig davon, auf welcher Seite der Transaktion Sie stehen.

Einschränkungen bei mehreren Lösungen

Die XIRR-Funktion von Qlik wird durch die folgende Gleichung definiert, in der der Wert `Rate` aufgelöst wird:

```
XNPV(Rate, pmt, date) = 0
```

Manchmal kann diese Gleichung mehr als eine Lösung haben. Dies ist als „Problem mit mehreren internen Zinsfüßen“ bekannt und wird durch einen nicht normalen Geldfluss (auch als unkonventioneller Geldfluss bezeichnet) verursacht. Das folgende Ladeskript zeigt ein Beispiel:

```
Cashflow:
LOAD * inline [
Date|Payments
```

```
2021-01-01|-200
2022-01-01|500
2023-01-01|-250
] (delimiter is '|');
```

In diesem Beispiel gibt es eine negative und eine positive Lösung (`Rate` = -0.3 und `Rate` = 0.8). **XIRR()** gibt 0.8 zurück.

Wenn die XIRR-Funktion von Qlik nach einer Lösung sucht, beginnt sie bei `Rate` = 0 und erhöht den Zinsfuß schrittweise, bis sie eine Lösung findet. Wenn es mehr als eine positive Lösung gibt, gibt die Funktion die erste gefundene Lösung zurück. Wenn keine positive Lösung gefunden wird, setzt die Funktion `Rate` auf Null zurück und beginnt, in negativer Richtung nach einer Lösung zu suchen.

Beachten Sie, dass es bei „normalem“ Geldfluss garantiert nur eine Lösung gibt. „Normaler“ Geldfluss bedeutet, dass sich alle Zahlungen mit dem gleichen Vorzeichen (positiv oder negativ) in einer fortlaufenden Gruppe befinden.

Siehe auch:

- *IRR - Diagrammfunktion (page 385)*
- *Aggr - Diagrammfunktion (page 566)*
- XIRR-Funktion – Behebung und Aktualisierung

XNPV

Die Funktion **XNPV()** gibt den aggregierten Nettobarwert einer Reihe geplanter (nicht unbedingt regelmäßig erfolgender) Geldflüsse zurück, repräsentiert durch die Wertepaare in **pmt** und **date**. Alle Beträge werden auf ein 365-Tage-Jahr hochgerechnet.

Syntax:

```
XNPV(discount_rate, pmt, date)
```

Rückgabe Datentyp: numerisch

Standardmäßig wird das Ergebnis als Währung formatiert.

Die Formel zum Berechnen von XNPV wird unten gezeigt:

XNPV-Aggregierungsformel

$$XNPV = \sum_{i=1}^{n} \frac{P_i}{(1+rate)^{(di-d1)/365}}$$

Dabei gilt:

- P_i = eingehende und ausgehende Nettozahlungsströme während eines einzelnen Zeitraums `i`
- d_1 = das erste Zahlungsdatum

- d_i = das ite Zahlungsdatum
- rate = Diskontierungsrate

Der Nettobarwert (Net Present Value, NPV) dient zum Berechnen des aktuellen Gesamtwerts von zukünftigen Geldflüssen bei einer bestimmten Diskontierungsrate. Zum Berechnen des XNPV müssen wir zukünftige Geldflüsse mit entsprechenden Datumswerten schätzen. Danach wenden wir für jede Zahlung die Gesamtdiskontierungsrate basierend auf dem Zahlungsdatum an.

Die XNPV-Aggegierung über eine Reihe von Zahlungen gleicht einer Summenaggregierung über diese Zahlungen. Der Unterschied besteht darin, dass jeder Betrag abgeändert („diskontiert“) wird – abhängig von der gewählten Diskontierungsrate (ähnlich einem Zinssatz) und abhängig davon, wie weit in der Zukunft die Zahlung liegt. Wenn für XNPV der Parameter **discount_rate** auf Null festgesetzt ist, entspricht XNPV einer Summenoperation (die Zahlungen werden vor dem Summieren nicht geändert). Generell gilt: Je näher **discount_rate** bei null liegt, desto stärker gleicht das XNPV-Ergebnis dem einer Summenaggregierung.

Argumente

Argument	Beschreibung
discount_rate	**discount_rate** ist die jährliche Rate, um die die Zahlungen diskontiert werden sollen. Ein Wert von 0,1 gibt eine Diskontierungsrate von 10 % an.
pmt	Zahlungen. Die Formel oder das Feld mit den Cashflows, die den Zahlungszeitpunkten in **date** entsprechen. Positive Werte gelten als Zahlungseingänge, negative Werte als Zahlungsausgänge. ***XNPV()*** *diskontiert den anfänglichen Geldfluss nicht, da diese immer am Startdatum stattfindet. Anschließende Zahlungen werden auf ein 365-Tage-Jahr hochgerechnet. Das ist anders als bei* ***NPV()****, wo auch die erste Zahlung diskontiert wird.*
date	Die Formel oder das Feld mit den Zeitpunkten, die den Cashflow-Zahlungen in **pmt** entsprechen. Der erste Wert wird als Startdatum für die Berechnung des Versatzes für zukünftige Geldflüsse verwendet.

Bei der Arbeit mit dieser Funktion gelten die folgenden Einschränkungen:

- Textwerte, NULL-Werte und fehlende Werte in einem oder beiden Teilen eines Wertepaars führen dazu, dass das Wertepaar ignoriert wird.

Verwendung

- XNPV() wird in der Finanzmodellierung zum Berechnen des Nettobarwerts (Net Present Value, NPV) einer Investitionsmöglichkeit verwendet.
- Aufgrund seiner höheren Präzision wird XNPV bei allen Typen von Finanzmodellen gegenüber NPV bevorzugt.

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Beispiel 1 – Einmalige Zahlung (Skript)

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz für ein Projekt und dessen Geldfluss für ein Jahr wird in eine Tabelle namens `CashFlow` geladen. Das anfängliche Datum für die Berechnung ist auf den 1. Juli 2022 mit einem Nettogeldfluss von 0 festgelegt. Nach einem Jahr findet ein Geldfluss von $1000 statt.
- Ein Resident Load aus der Tabelle `CashFlow` wird verwendet, um das `XNPV`-Feld für das Projekt in der Tabelle `XNPV` zu berechnen.
- Eine feste Diskontierungsrate von 10 % (0,1) wird in der XNPV-Berechnung verwendet.
- Ein `Group By`-Befehl wird verwendet, um alle Zahlungen für das Projekt zu gruppieren.

Ladeskript

```
CashFlow:
Load
*
Inline
[
PrjId,Dates,Values
1,'07/01/2022',0
1,'07/01/2023',1000
];

XNPV:
Load
       PrjId,
       XNPV(0.1,Values,Dates) as XNPV //Discount Rate of 10%
```

```
Resident CashFlow
Group By PrjId;
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- PrjId
- XNPV

Ergebnistabelle

PrjId	XNPV
1	$909.09

Entsprechend der Formel ist der XNPV-Wert für den ersten Datensatz 0 und für den zweiten Datensatz $909,09. Somit beträgt der gesamte XNPV $909,09.

Beispiel 2 – Mehrere Zahlungen (Skript)

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz für ein Projekt und dessen Geldfluss für ein Jahr wird in eine Tabelle namens CashFlow geladen.
- Ein Resident Load aus der Tabelle CashFlow wird verwendet, um das XNPV-Feld für das Projekt in der Tabelle XNPV zu berechnen.
- Eine feste Diskontierungsrate von 10 % (0,1) wird in der XNPV-Berechnung verwendet.
- Ein Group By-Befehl wird verwendet, um alle Zahlungen für das Projekt zu gruppieren.

Ladeskript

```
CashFlow:
Load
*
Inline
[
PrjId,Dates,Values
1,'07/01/2022',0
1,'07/01/2024',500
1,'07/01/2023',1000
];

XNPV:
Load
```

```
       PrjId,
       XNPV(0.1,Values,Dates) as XNPV //Discount Rate of 10%
Resident CashFlow
Group By PrjId;
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- PrjId
- XNPV

Ergebnistabelle

PrjId	XNPV
1	$1322.21

In diesem Beispiel wird am Ende des ersten Jahres eine Zahlung von $1000 und am Ende des zweiten Jahres eine Zahlung von $500 erhalten. Mit einer Diskontierungsrate von 10 % pro Zeitraum entspricht der effektive XNPV $1322,21.

Beachten Sie, dass nur die erste Datenzeile auf das Basisdatum für Berechnungen verweisen sollte. Für die restlichen Zeilen ist die Reihenfolge nicht wichtig, da der Datumsparameter zum Berechnen des verstrichenen Zeitraums verwendet wird.

Beispiel 3 – Mehrere Zahlungen und unregelmäßige Geldflüsse (Skript)

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Diskontierungsraten für zwei Projekte werden in eine Tabelle namens Project geladen.
- Es erfolgen Geldflüsse für mehrere Zeiträume für jedes Projekt nach Projekt-ID und Datum. Das Feld Dates wird verwendet, um die Dauer zu berechnen, für die die Diskontierungsrate auf den Geldfluss angewendet wird. Abgesehen vom ersten Datensatz (anfänglicher Geldfluss und Datum) ist die Reihenfolge der Datensätze nicht wichtig, und eine Änderung dürfte keine Auswirkung auf die Berechnungen haben.
- Mit einer Kombination aus NoConcatenate, Resident Loads und der Funktion Left Join wird eine temporäre Tabelle tmpNPV erstellt, in der die Datensätze aus den Tabellen Project und CashFlow in einer flachen Tabelle kombiniert werden. In dieser Tabelle werden Diskontierungsraten für jeden Geldfluss wiederholt.
- Ein Resident Load aus der Tabelle tmpNPV wird verwendet, um das Feld XNPV für jedes Projekt in der Tabelle XNPV zu berechnen.

- Die Diskontierungsrate mit einem einzelnen Wert für jedes Projekt wird anhand der Funktion `only()` abgerufen und in der XNPV-Berechnung für jedes Projekt verwendet.
- Ein `Group By`-Befehl, der nach Projekt-ID gruppiert, wird zum Gruppieren aller Zahlungen für jedes Projekt und der entsprechenden Datumswerte verwendet.
- Um zu verhindern, dass synthetische oder redundante Daten in das Datenmodell geladen werden, wird die Tabelle `tmpXNPV` am Ende des Skripts verworfen.

Ladeskript

```
Project:
Load * inline [
PrjId,Discount_Rate
1,0.1
2,0.15
];

CashFlow:
Load
*
Inline
[
PrjId,Dates,Values
1,'07/01/2021',0
1,'07/01/2022',1000
1,'07/01/2023',1000
2,'07/01/2020',0
2,'07/01/2023',500
2,'07/01/2024',1000
2,'07/01/2022',500
];

tmpXNPV:
NoConcatenate Load *
Resident Project;
Left Join
Load *
Resident CashFlow;

XNPV:
Load
       PrjId,
       XNPV(Only(Discount_Rate),Values,Dates) as XNPV //Discount Rate will be 10% for Project 1 and
15% for Project 2
Resident tmpXNPV
Group By PrjId;

Drop table tmpXNPV;
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- PrjId
- XNPV

Ergebnistabelle

PrjId	XNPV
1	$1735.54
2	$278.36

Projekt-ID 1 hat einen anfänglichen Geldfluss von $0 am Juli 1. 2021. Es sollen zwei Zahlungen von $1000 am Ende von zwei aufeinanderfolgenden Jahren mit einer Diskontierungsrate von 10 % pro Zeitraum erhalten werden. Daher entspricht der effektive XNPV $1735,54.

Projekt-ID 2 hat einen anfänglichen ausgehenden Geldfluss von $1000 (daher das negative Vorzeichen) am 1. Juli 2020. Nach zwei Jahren wird eine Zahlung von $500 erwartet. Nach drei Jahren wird eine weitere Zahlung von $500 erwartet. Abschließend wird am 1. Juli 2024 eine Zahlung von $1000 erwartet. Mit einer Diskontierungsrate von 15 % entspricht der effektive XNPV $278,36.

Siehe auch:

- *Drop table (page 155)*
- *group by (page 166)*
- *Join (page 75)*
- *Max (page 349)*
- *NoConcatenate (page 93)*
- *NPV - Diagrammfunktion (page 394)*
- *Only (page 359)*

XNPV - Diagrammfunktion

XNPV() liefert den aggregierten Nettobarwert einer Reihe (nicht notwendigerweise regelmäßig erfolgenden) Cashflows, repräsentiert durch die Wertepaare in den Formeln von **pmt** und **date**, iteriert nach den Dimensionen des Diagramms. Alle Beträge werden auf ein 365-Tage-Jahr hochgerechnet.

Syntax:

```
XNPV([TOTAL [<fld{,fld}>]] discount_rate, pmt, date)
```

Rückgabe Datentyp: numerisch

Standardmäßig wird das Ergebnis als Währung formatiert.

Die Formel zum Berechnen von XNPV wird unten gezeigt:

XNPV-Aggregierungsformel

$$XNPV = \sum_{i=1}^{n} \frac{P_i}{(1+rate)^{(di-d1)/365}}$$

Dabei gilt:

- P_i = eingehende und ausgehende Nettozahlungsströme während eines einzelnen Zeitraums `i`
- d_1 = das erste Zahlungsdatum
- d_i = das `i`te Zahlungsdatum
- `rate` = Diskontierungsrate

Der Nettobarwert (Net Present Value, NPV) dient zum Berechnen des aktuellen Gesamtwerts von zukünftigen Geldflüssen bei einer bestimmten Diskontierungsrate. Zum Berechnen des XNPV müssen wir zukünftige Geldflüsse mit entsprechenden Datumswerten schätzen. Danach wenden wir für jede Zahlung die Gesamtdiskontierungsrate basierend auf dem Zahlungsdatum an.

Die XNPV-Aggegierung über eine Reihe von Zahlungen gleicht einer Summenaggregierung über diese Zahlungen. Der Unterschied besteht darin, dass jeder Betrag abgeändert („diskontiert“) wird – abhängig von der gewählten Diskontierungsrate (ähnlich einem Zinssatz) und abhängig davon, wie weit in der Zukunft die Zahlung liegt. Wenn für XNPV der Parameter **discount_rate** auf Null festgesetzt ist, entspricht XNPV einer Summenoperation (die Zahlungen werden vor dem Summieren nicht geändert). Generell gilt: Je näher **discount_rate** bei null liegt, desto stärker gleicht das XNPV-Ergebnis dem einer Summenaggregierung.

Argumente

Argument	Beschreibung
discount_rate	**discount_rate** ist die jährliche Rate, um die die Zahlungen diskontiert werden sollen. Ein Wert von 0,1 gibt eine Diskontierungsrate von 10 % an.
pmt	Zahlungen. Die Formel oder das Feld mit den Cashflows, die den Zahlungszeitpunkten in **date** entsprechen. Positive Werte gelten als Zahlungseingänge, negative Werte als Zahlungsausgänge. ***XNPV()*** *diskontiert den anfänglichen Geldfluss nicht, da diese immer am Startdatum stattfindet. Anschließende Zahlungen werden auf ein 365-Tage-Jahr hochgerechnet. Das ist anders als bei* ***NPV()****, wo auch die erste Zahlung diskontiert wird.*
date	Die Formel oder das Feld mit den Zeitpunkten, die den Cashflow-Zahlungen in **pmt** entsprechen. Der erste Wert wird als Startdatum für die Berechnung des Versatzes für zukünftige Geldflüsse verwendet.

Argument	Beschreibung
TOTAL	Der Zusatz **TOTAL** vor der Funktion bewirkt, dass die Berechnung über alle ausgewählten bzw. wählbaren Werte erfolgt, und nicht nur über diejenigen, die zu dem Wert der aktuellen Dimension zählen. Die Dimensionen des Diagramms werden also nicht berücksichtigt. Mit **TOTAL [<fld {.fld}>]**, wobei auf den Zusatz **TOTAL** eine Liste aus mindestens einem Feldnamen (d. h. einer Teilmenge der Diagrammdimensionsvariablen) folgt, erstellen Sie eine Teilmenge aller möglichen Werte.

Bei der Arbeit mit dieser Funktion gelten die folgenden Einschränkungen:

- **discount_rate**, **pmt** und **date** dürfen keine Aggregierungsfunktionen enthalten, es sei denn, die inneren Aggregierungen enthalten den Qualifizierer **TOTAL** oder **ALL**. Für komplexere verschachtelte Aggregierungen verwenden Sie die erweiterte Funktion **Aggr** in Verbindung mit einer angegebenen Dimension.
- Textwerte, NULL-Werte und fehlende Werte in einem oder beiden Teilen eines Wertepaars führen dazu, dass das Wertepaar ignoriert wird.

Verwendung

- `XNPV()` wird in der Finanzmodellierung zum Berechnen des Nettobarwerts (Net Present Value, NPV) einer Investitionsmöglichkeit verwendet.
- Aufgrund seiner höheren Präzision wird XNPV bei allen Typen von Finanzmodellen gegenüber NPV bevorzugt.

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Beispiel

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz enthält Geldflusstransaktionen.
- Die Informationen sind in einer Tabelle mit dem Namen `Cashflow` gespeichert.

Ladeskript

```
Cashflow:
LOAD 2013 as Year, * inline [
Date|Payments
2013-01-01|-10000
2013-03-01|3000
2013-10-30|4200
2014-02-01|6800
] (delimiter is '|');
```

Ergebnisse

Gehen Sie folgendermaßen vor:

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Berechnungen als Kennzahl hinzu:

```
=XNPV(0.09, Payments, Date)
```

Ergebnistabelle

=XNPV(0.09, Payments, Date)
$3062.49

Siehe auch:

NPV - Diagrammfunktion (page 394)
Aggr - Diagrammfunktion (page 566)

Statistische Aggregierungsfunktionen

Jede Funktion wird nach der Übersicht genauer beschrieben. Sie können auch auf den Funktionsnamen in der Syntax klicken, um direkt auf die Details zu der spezifischen Funktion zuzugreifen.

Statistische Aggregierungsfunktionen im Datenladeskript

In Skripten können folgende statistische Aggregierungsfunktionen verwendet werden.

Avg
Avg() liefert den Durchschnittswert der aggregierten Daten in der Formel über mehrere Datensätze, wie in einer Bedingung **group by** definiert wurde.

```
Avg  ([distinct] expression)
```

Correl

Correl() liefert den Korrelationskoeffizienten für eine Reihe von Wertepaaren, die durch x-expression und y-expression definiert sind, über die in der **group by**-Bedingung bezeichneten Datensätze.

```
Correl (x-expression, y-expression)
```

Fractile

Fractile() liefert den Wert, der dem inklusiven Fraktil (Quantil) der aggregierten Daten in der Formel über mehrere Datensätze entspricht, wie in einer **group by**-Klausel definiert wurde.

```
Fractile (expression, fractile)
```

FractileExc

FractileExc() liefert den Wert, der dem exklusiven Fraktil (Quantil) der aggregierten Daten in der Formel über mehrere Datensätze entspricht, wie in einer **group by**-Klausel definiert wurde.

```
FractileExc (expression, fractile)
```

Kurtosis

Kurtosis() liefert die Kurtosis der Daten in der Formel über mehrere Datensätze, wie in einer Bedingung **group by** definiert wurde.

```
Kurtosis  ([distinct ] expression )
```

LINEST_B

LINEST_B() liefert den y-Achsenabschnitt (Regressionskonstante) b in der durch die Gleichung y=mx+b bestimmten linearen Regression für eine Reihe von Koordinaten, die durch x-expression und y-expression definiert sind, über die im **group by**-Zusatz bezeichneten Datensätze.

```
LINEST_B (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_df

LINEST_DF() liefert die aggregierten Freiheitsgrade in der durch die Gleichung y=mx+b bestimmten linearen Regression für eine Reihe von Koordinaten, die durch x-expression und y-expression definiert sind, aggregiert über die in einer **group by**-Bedingung bezeichneten Datensätze.

```
LINEST_DF  (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_f

Diese Skriptfunktion liefert den F-Wert ($r^2/(1-r^2)$), der durch die Gleichung y=mx+b bestimmten linearen Regression für eine Reihe von Koordinaten, die durch x-expression und y-expression definiert sind, aggregiert über eine durch den **group by**-Zusatz festgelegte Anzahl von Datensätzen.

```
LINEST_F (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_m

LINEST_M() liefert den m-Wert (Steigung) in der durch die Gleichung y=mx+b bestimmten linearen Regression für eine Reihe von Koordinaten, die durch x-expression und y-expression definiert sind, aggregiert über die in einer **group by**-Bedingung bezeichneten Datensätze.

```
LINEST_M  (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_r2

LINEST_R2() liefert den aggregierten Wert r^2 (Bestimmtheitsmaß) einer durch die Gleichung y=mx+b bestimmten linearen Regression für eine Reihe von Koordinaten, die durch x-expression und y-expression definiert sind, aggregiert über eine Anzahl von Datensätzen, die durch eine **group by**-Bedingung definiert sind.

```
LINEST_R2 (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_seb

LINEST_SEB() liefert den Standardfehler des b-Werts in der durch die Gleichung y=mx+b bestimmten linearen Regression für eine Reihe von Koordinaten, die durch x-expression und y-expression definiert sind, über die im **group by**-Zusatz bezeichneten Datensätze.

```
LINEST_SEB  (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_sem

LINEST_SEM() liefert den Standardfehler des m-Werts in der durch die Gleichung y=mx+b bestimmten linearen Regression für eine Reihe von Koordinaten, die durch x-expression und y-expression definiert sind, über die im **group by**-Zusatz bezeichneten Datensätze.

```
LINEST_SEM  (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_sey

LINEST_SEY() liefert den Standardfehler des geschätzten y-Wertes in der durch die Gleichung y=mx+b bestimmten linearen Regression für eine Reihe von Koordinaten, die durch x-expression und y-expression definiert sind, über die im **group by**-Zusatz bezeichneten Datensätze.

```
LINEST_SEY  (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_ssreg

LINEST_SSREG() liefert die durch die Regression erklärte Varianz − Regression Sum of Squares − der durch die Gleichung y=mx+b bestimmten linearen Regression für eine Reihe von Koordinaten, die durch x-expression und y-expression definiert sind, aggregiert über eine Anzahl von Datensätzen, die durch eine **group by**-Bedingung definiert sind.

```
LINEST_SSREG  (y-expression, x-expression [, y0 [, x0 ]])
```

Linest_ssresid

LINEST_SSRESID() liefert die durch die residuale oder nicht erklärte Varianz, Residual Sum of Squares, der durch die Gleichung y=mx+b bestimmten linearen Regression für eine Reihe von Koordinaten, die durch x-expression und y-expression definiert sind, über die im **group by**-Zusatz bezeichneten Datensätze.

```
LINEST_SSRESID  (y-expression, x-expression [, y0 [, x0 ]])
```

Median

Median() liefert den aggregierten Median der Werte in der Formel über mehrere Datensätze, wie durch eine **group by**-Bedingung definiert wurde.

```
Median (expression)
```

Skew

Skew() liefert die Schiefe der Formel über die im **group by**-Zusatz bezeichneten Datensätze.

```
Skew ([ distinct] expression)
```

Stdev

Stdev() liefert die Standardabweichung der in der Formel über mehrere Datensätze angegebenen Werte, wie in einer Bedingung **group by** definiert wurde.

```
Stdev ([distinct] expression)
```

Sterr

Sterr() liefert den Standardfehler (stdev/sqrt(n)) für eine Wertemenge einer Formel über die in der **group by**-Bedingung bezeichneten Datensätze.

```
Sterr  ([distinct] expression)
```

STEYX

STEYX() liefert den Standardfehler des geschätzten y-Werts für jeden x-Wert in der Regression für eine Reihe von Wertepaaren, die durch x-expression und y-expression definiert sind, über die im **group by**-Zusatz bezeichneten Datensätze.

```
STEYX  (y-expression, x-expression)
```

Statistische Aggregierungsfunktionen in Diagrammformeln

Die folgenden statistischen Aggregierungsfunktionen können in Diagrammen verwendet werden.

Avg

Avg() liefert den Mittelwert der Werte der Formel oder des Felds aggregiert nach den Dimensionen des Diagramms.

```
Avg - Diagrammfunktion({[SetExpression] [DISTINCT] [TOTAL [<fld{, fld}>]]}
expr)
```

Correl

Correl() liefert den Korrelationskoeffizienten für zwei Datensätze. Die Korrelationsfunktion ist eine Kennzahl der Relation zwischen den Datensätzen und ist für Wertepaare (x,y) aggregiert nach den Dimensionen des Diagramms.

```
Correl - Diagrammfunktion({[SetExpression] [TOTAL [<fld {, fld}>]]} value1,
value2 )
```

Fractile

Fractile() sucht nach dem Wert, der dem inklusiven Fraktil (Quantil) der aggregierten Daten in dem Bereich entspricht, der von der Formel über die Dimensionen des Diagramms vorgegeben ist.

```
Fractile - Diagrammfunktion({[SetExpression] [TOTAL [<fld {, fld}>]]} expr,
fraction)
```

FractileExc

FractileExc() sucht nach dem Wert, der dem exklusiven Fraktil (Quantil) der aggregierten Daten in dem Bereich entspricht, der von der Formel über die Dimensionen des Diagramms vorgegeben ist.

```
FractileExc - Diagrammfunktion({[SetExpression] [TOTAL [<fld {, fld}>]]}
expr, fraction)
```

Kurtosis

Kurtosis() sucht nach der Kurtosis des in der Formel oder im Feld aggregierten Bereichs von Daten, die über die Dimensionen des Diagramms iteriert wurden.

```
Kurtosis - Diagrammfunktion({[SetExpression] [DISTINCT] [TOTAL [<fld{,
fld}>]]} expr)
```

LINEST_b

LINEST_B() liefert den Wert b (y-Achsenabschnitt) in der durch die Gleichung y=mx+b bestimmten linearen Regression für eine Reihe von Koordinaten, die durch **x_value** und **y_value** definiert sind, aggregiert nach den Dimensionen des Diagramms.

```
LINEST_R2 - Diagrammfunktion({[SetExpression] [TOTAL [<fld{ ,fld}>]] }y_
value, x_value[, y0_const[, x0_const]])
```

LINEST_df

LINEST_DF() liefert die Freiheitsgrade der durch die Gleichung y=mx+b bestimmten linearen Regression für eine Reihe von Koordinaten, die durch **x_value** und **y_value** definiert sind, aggregiert nach den Dimensionen des Diagramms.

```
LINEST_DF - Diagrammfunktion({[SetExpression] [TOTAL [<fld{, fld}>]]} y_
value, x_value [, y0_const [, x0_const]])
```

LINEST_f

LINEST_F() liefert den F-Wert (r2/(1-r2)) der durch die Gleichung y=mx+b bestimmten linearen Regression für eine Reihe von Koordinaten, die durch **x_value** und **y_value** definiert sind, aggregiert nach den Dimensionen des Diagramms.

```
LINEST_F - Diagrammfunktion({[SetExpression] [TOTAL[<fld{, fld}>]]} y_value,
x_value [, y0_const [, x0_const]])
```

LINEST_m

LINEST_M() liefert den m-Wert (Steigung) in der durch die Gleichung y=mx+b bestimmten linearen Regression für eine Reihe von Koordinaten, die durch **x_value** und **y_value** definiert sind, aggregiert nach den Dimensionen des Diagramms.

```
LINEST_M - Diagrammfunktion({[SetExpression] [TOTAL[<fld{, fld}>]]} y_value,
x_value [, y0_const [, x0_const]])
```

LINEST_r2

LINEST_R2() liefert den r2-Wert (Determinationskoeffizient) in der durch die Gleichung y=mx+b bestimmten linearen Regression für eine Reihe von Koordinaten, die durch **x_value** und **y_value** definiert sind, aggregiert nach den Dimensionen des Diagramms.

```
LINEST_R2 - Diagrammfunktion({[SetExpression] [TOTAL [<fld{ ,fld}>]] }y_
value, x_value[, y0_const[, x0_const]])
```

LINEST_seb

LINEST_SEB() liefert den Standardfehler des b-Werts in der durch die Gleichung y=mx+b bestimmten linearen Regression für eine Reihe von Koordinaten, die durch **x_value** und **y_value** definiert sind, aggregiert nach den Dimensionen des Diagramms.

```
LINEST_SEB - Diagrammfunktion({[SetExpression] [TOTAL [<fld{ ,fld}>]] }y_
value, x_value[, y0_const[, x0_const]])
```

LINEST_sem

LINEST_SEM() liefert den Standardfehler des m-Werts in der durch die Gleichung y=mx+b bestimmten linearen Regression für eine Reihe von Koordinaten, die durch **x_value** und **y_value** definiert sind, aggregiert nach den Dimensionen des Diagramms.

```
LINEST_SEM - Diagrammfunktion([{set_expression}][ distinct ] [total [<fld
{,fld}>] ] y-expression, x-expression [, y0 [, x0 ]] )
```

LINEST_sey

LINEST_SEY() liefert den Standardfehler des geschätzten y-Werts in der durch die Gleichung y=mx+b bestimmten linearen Regression für eine Reihe von Koordinaten, die durch **x_value** und **y_value** definiert sind, aggregiert nach den Dimensionen des Diagramms.

```
LINEST_SEY - Diagrammfunktion({[SetExpression] [TOTAL [<fld{ ,fld}>]] }y_
value, x_value[, y0_const[, x0_const]])
```

LINEST_ssreg

LINEST_SSREG() liefert die durch die Regression erklärte Varianz, Regression Sum of Squares, in der durch die Gleichung y=mx+b bestimmten linearen Regression für eine Reihe von Koordinaten, die durch **x_value** und **y_value** definiert sind, aggregiert nach den Dimensionen des Diagramms.

```
LINEST_SSREG - Diagrammfunktion({[SetExpression] [TOTAL [<fld{ ,fld}>]] }y_
value, x_value[, y0_const[, x0_const]])
```

LINEST_ssresid

LINEST_SSRESID() liefert die residuale oder nicht erklärte Varianz, Residual Sum of Squares, der durch die Gleichung y=mx+b bestimmten linearen Regression für eine Reihe von Koordinaten, die durch die Formeln **x_value** und **y_value** definiert sind, aggregiert nach den Dimensionen des Diagramms.

```
LINEST_SSRESID - DiagrammfunktionLINEST_SSRESID() liefert die residuale oder
nicht erklärte Varianz, Residual Sum of Squares, der durch die Gleichung
y=mx+b bestimmten linearen Regression für eine Reihe von Koordinaten, die
durch die Formeln x_value und y_value definiert sind, aggregiert nach den
Dimensionen des Diagramms.  LINEST_SSRESID([{SetExpression}] [DISTINCT]
[TOTAL [<fld{, fld}>]] y_value, x_value[, y0_const[, x0_const]])
numerisch ArgumenteArgumentBeschreibungy_valueDie Formel oder das Feld mit
dem Datenbereich y, der angegeben werden soll.x_valueDie Formel oder das Feld
mit dem Datenbereich x, der angegeben werden soll.y0, x0Der optionale
Parameter y0definiert, bei welchem Wert die Regressionsgerade die y-Achse
schneidet. Die optionalen Parameter y0 und x0 definieren einen
Koordinatenpunkt, durch den die Regressionsgerade geht. Sofern die Parameter
```

y0 und x0 nicht definiert sind, erfordert die Berechnung zwei gültige Datenpaare. Wenn y0 und x0 definiert sind, reicht ein Datenpaar. SetExpressionStandardmäßig berechnet sich die Aggregierungsfunktion über alle wählbaren Werte. Alternativ können Sie die der Berechnung zugrunde liegenden Werte über die Auswahlformel bestimmen. DISTINCTDer Zusatz DISTINCT vor den Funktionsargumenten bewirkt, dass bei der Auswertung der Funktionsargumente entstehende Duplikate nicht berücksichtigt werden. TOTALDer Zusatz TOTAL vor der Funktion bewirkt, dass die Berechnung über alle ausgewählten bzw. wählbaren Werte erfolgt, und nicht nur über diejenigen, die zu dem Wert der aktuellen Dimension zählen. Die Dimensionen des Diagramms werden also nicht berücksichtigt. Mit TOTAL [<fld {.fld}>], wobei auf den Zusatz TOTAL eine Liste aus mindestens einem Feldnamen (d. h. einer Teilmenge der Diagrammdimensionsvariablen) folgt, erstellen Sie eine Teilmenge aller möglichen Werte.Der optionale Parameter y0definiert, bei welchem Wert die Regressionsgerade die y-Achse schneidet. Die optionalen Parameter y0 und x0 definieren einen Koordinatenpunkt, durch den die Regressionsgerade geht. Die Parameter der Aggregierungsfunktion dürfen keine anderen Aggregierungsfunktionen enthalten, es sei denn, diese inneren Aggregierungen enthalten den Qualifizierer TOTAL. Für komplexere verschachtelte Aggregierungen verwenden Sie die erweiterte Funktion Aggr in Verbindung mit einer angegebenen Dimension. Textwerte, NULL-Werte und fehlende Werte in einem oder beiden Teilen eines Wertepaars führen dazu, dass das Wertepaar ignoriert wird. An example of how to use linest functionsavg ({[SetExpression] [**TOTAL** [<fld{ ,fld}>]] }y_value, x_value[, y0_const[, x0_const]]**)**

Median

Median() liefert den Median des Bereichs von Werten, die in der Formel über die Dimensionen des Diagramms aggregiert wurden.

```
Median - Diagrammfunktion({[SetExpression] [TOTAL [<fld{, fld}>]]} expr)
```

MutualInfo

MutualInfo berechnet die gegenseitigen Informationen zwischen zwei Feldern oder zwischen aggregierten Werten in **Aggr()**.

```
MutualInfo - Diagrammfunktion{[SetExpression] [DISTINCT] [TOTAL target,
driver [, datatype [, breakdownbyvalue [, samplesize ]]]]
```

Skew

Skew() liefert die Schiefe der Werte der Formel oder des Felds aggregiert nach den Dimensionen des Diagramms.

```
Skew - Diagrammfunktion{[SetExpression] [DISTINCT] [TOTAL [<fld{ ,fld}>]]}
expr)
```

Stdev

Stdev() sucht nach der Standardabweichung des in der Formel oder im Feld aggregierten Bereichs von Daten, die über die Dimensionen des Diagramms iteriert wurden.

```
Stdev - Diagrammfunktion({[SetExpression] [DISTINCT] [TOTAL [<fld{, fld}>]]}
expr)
```

Sterr

Sterr() liefert den Wert des Standardfehlers des Mittels (stdev/sqrt(n)) für die Werte, die in der Formel über die Dimensionen des Diagramms aggregiert wurden.

```
Sterr - Diagrammfunktion({[SetExpression] [DISTINCT] [TOTAL[<fld{, fld}>]]}
expr)
```

STEYX

STEYX() liefert den Standardfehler der geschätzten y-Werte für jeden x-Wert in einer linearen Regression für eine Reihe von Koordinaten, die durch **y_value** und **x_value** definiert sind.

```
STEYX - Diagrammfunktion{[SetExpression] [TOTAL [<fld{, fld}>]]} y_value, x_
value)
```

Avg

Avg() liefert den Durchschnittswert der aggregierten Daten in der Formel über mehrere Datensätze, wie in einer Bedingung **group by** definiert wurde.

Syntax:

```
Avg([DISTINCT] expr)
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
expr	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.
DISTINCT	Ist der Formel das Wort **distinct** vorangestellt, werden Dubletten nicht berücksichtigt.

Beispiele und Ergebnisse:

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Fügen Sie einem Arbeitsblatt in Ihrer App dann die Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um das Ergebnis anzuzeigen.

Ergebnisdaten

Beispiel	Ergebnis
Temp: crosstable (Month, Sales) load * inline [Customer|Jan|Feb|Mar||Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec Astrida|46|60|70|13|78|20|45|65|78|12|78|22 Betacab|65|56|22|79|12|56|45|24|32|78|55|15 Canutility|77|68|34|91|24|68|57|36|44|90|67|27 Divadip|36|44|90|67|27|57|68|47|90|80|94] (delimiter is '|'); Avg1: LOAD Customer, Avg(Sales) as MyAverageSalesByCustomer Resident Temp Group By Customer;	Customer MyAverageSalesByCustomer Astrida 48.916667 Betacab 44.916667 Canutility 56.916667 Divadip 63.083333 Das lässt sich im Arbeitsblatt überprüfen, indem eine Tabelle mit der Kennzahl erstellt wird . Sum(Sales)/12
Vorgabe: Die Tabelle **Temp** wird wie im vorherigen Beispiel geladen: LOAD Customer,Avg(DISTINCT Sales) as MyAvgSalesDistinct Resident Temp Group By Customer;	Customer MyAverageSalesByCustomer Astrida 43.1 Betacab 43.909091 Canutility 55.909091 Divadip 61 Nur verschiedene Werte werden gezählt. Teilen Sie den Gesamtwert durch die Anzahl der duplizierten Werte.

Avg - Diagrammfunktion

Avg() liefert den Mittelwert der Werte der Formel oder des Felds aggregiert nach den Dimensionen des Diagramms.

Syntax:

```
Avg([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr)
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
expr	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.

Argument	Beschreibung
SetExpression	Standardmäßig berechnet sich die Aggregierungsfunktion über alle wählbaren Werte. Alternativ können Sie die der Berechnung zugrunde liegenden Werte über die Auswahlformel bestimmen.
DISTINCT	Der Zusatz **DISTINCT** vor den Funktionsargumenten bewirkt, dass bei der Auswertung der Funktionsargumente entstehende Duplikate nicht berücksichtigt werden.
TOTAL	Der Zusatz **TOTAL** vor der Funktion bewirkt, dass die Berechnung über alle ausgewählten bzw. wählbaren Werte erfolgt, und nicht nur über diejenigen, die zu dem Wert der aktuellen Dimension zählen. Die Dimensionen des Diagramms werden also nicht berücksichtigt. Mit **TOTAL [<fld {.fld}>]**, wobei auf den Zusatz **TOTAL** eine Liste aus mindestens einem Feldnamen (d. h. einer Teilmenge der Diagrammdimensionsvariablen) folgt, erstellen Sie eine Teilmenge aller möglichen Werte.

Beschränkungen:

Die Parameter der Aggregierungsfunktion dürfen keine anderen Aggregierungsfunktionen enthalten, es sei denn, diese inneren Aggregierungen enthalten den Qualifizierer **TOTAL**. Für komplexere verschachtelte Aggregierungen verwenden Sie die erweiterte Funktion **Aggr** in Verbindung mit einer angegebenen Dimension.

Beispiele und Ergebnisse:

Example table

Customer	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Astrida	46	60	70	13	78	20	45	65	78	12	78	22
Betacab	65	56	22	79	12	56	45	24	32	78	55	15
Canutility	77	68	34	91	24	68	57	36	44	90	67	27
Divadip	57	36	44	90	67	27	57	68	47	90	80	94

Funktionsbeispiele

Beispiel	Ergebnis
`Avg(Sales)`	Für eine Tabelle mit der Dimension `Customer` und der Kennzahl `Avg([Sales])` ist das Ergebnis 2566, wenn **Gesamtwerte** angezeigt werden.
`Avg([TOTAL (Sales))`	53,458333 für alle Werte von `Customer`, da durch den Zusatz TOTAL die Dimensionen nicht berücksichtigt werden.
`Avg(DISTINCT (Sales))`	51,862069 für den Gesamtwert, da durch den Zusatz Distinct in `Sales` für jeden `Customer` nur eindeutige Werte berechnet werden.

In Beispielen verwendete Daten:

```
Monthnames:
LOAD *, Dual(MonthText,MonthNumber) as Month INLINE [
```

```
MonthText, MonthNumber
Jan, 1
Feb, 2
Mar, 3
Apr, 4
May, 5
Jun, 6
Jul, 7
Aug, 8
Sep, 9
Oct, 10
Nov, 11
Dec, 12
];

Sales2013:
Crosstable (MonthText, Sales) LOAD * inline [
Customer|Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec
Astrida|46|60|70|13|78|20|45|65|78|12|78|22
Betacab|65|56|22|79|12|56|45|24|32|78|55|15
Canutility|77|68|34|91|24|68|57|36|44|90|67|27
Divadip|57|36|44|90|67|27|57|68|47|90|80|94
] (delimiter is '|');
```

Siehe auch:

Aggr - Diagrammfunktion (page 566)

Correl

Correl() liefert den Korrelationskoeffizienten für eine Reihe von Wertepaaren, die durch x-expression und y-expression definiert sind, über die in der **group by**-Bedingung bezeichneten Datensätze.

Syntax:

```
Correl(value1, value2)
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
value1, value2	Die Formeln oder Felder, welche die beiden Beispielgruppen enthalten, für die der Korrelationskoeffizient ermittelt werden soll.

Beschränkungen:

Textwerte, NULL-Werte und fehlende Werte in einem oder beiden Teilen eines Wertepaars führen dazu, dass das Wertepaar ignoriert wird.

Beispiele und Ergebnisse:

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Fügen Sie einem Arbeitsblatt in Ihrer App dann die Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um das Ergebnis anzuzeigen.

Ergebnisdaten

Beispiel	Ergebnis
Salary: Load *, 1 as Grp; LOAD * inline ["Employee name"|Gender|Age|Salary Aiden Charles|Male|20|25000 Brenda Davies|Male|25|32000 Charlotte Edberg|Female|45|56000 Daroush Ferrara|Male|31|29000 Eunice Goldblum|Female|31|32000 Freddy Halvorsen|Male|25|26000 Gauri Indu|Female|36|46000 Harry Jones|Male|38|40000 Ian Underwood|Male|40|45000 Jackie Kingsley|Female|23|28000] (delimiter is '|'); Correl1: LOAD Grp, Correl(Age,Salary) as Correl_ Salary Resident Salary Group By Grp;	In einer Tabelle mit der Dimension `Correl_Salary` wird das Ergebnis der Berechnung von Correl() im Datenladeskript angezeigt: 0,9270611

Correl - Diagrammfunktion

Correl() liefert den Korrelationskoeffizienten für zwei Datensätze. Die Korrelationsfunktion ist eine Kennzahl der Relation zwischen den Datensätzen und ist für Wertepaare (x,y) aggregiert nach den Dimensionen des Diagramms.

Syntax:

```
Correl([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] value1, value2 )
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
value1, value2	Die Formeln oder Felder, welche die beiden Beispielgruppen enthalten, für die der Korrelationskoeffizient ermittelt werden soll.
SetExpression	Standardmäßig berechnet sich die Aggregierungsfunktion über alle wählbaren Werte. Alternativ können Sie die der Berechnung zugrunde liegenden Werte über die Auswahlformel bestimmen.
DISTINCT	Der Zusatz **DISTINCT** vor den Funktionsargumenten bewirkt, dass bei der Auswertung der Funktionsargumente entstehende Duplikate nicht berücksichtigt werden.
TOTAL	Der Zusatz **TOTAL** vor der Funktion bewirkt, dass die Berechnung über alle ausgewählten bzw. wählbaren Werte erfolgt, und nicht nur über diejenigen, die zu dem Wert der aktuellen Dimension zählen. Die Dimensionen des Diagramms werden also nicht berücksichtigt. Mit **TOTAL [<fld {.fld}>]**, wobei auf den Zusatz **TOTAL** eine Liste aus mindestens einem Feldnamen (d. h. einer Teilmenge der Diagrammdimensionsvariablen) folgt, erstellen Sie eine Teilmenge aller möglichen Werte.

Beschränkungen:

Die Parameter der Aggregierungsfunktion dürfen keine anderen Aggregierungsfunktionen enthalten, es sei denn, diese inneren Aggregierungen enthalten den Qualifizierer **TOTAL**. Für komplexere verschachtelte Aggregierungen verwenden Sie die erweiterte Funktion **Aggr** in Verbindung mit einer angegebenen Dimension.

Textwerte, NULL-Werte und fehlende Werte in einem oder beiden Teilen eines Wertepaars führen dazu, dass das Wertepaar ignoriert wird.

Beispiele und Ergebnisse:

Funktionsbeispiele

Beispiel	Ergebnis
`Correl (Age, Salary)`	Für eine Tabelle mit der Dimension `Employee name` und der Kennzahl `Correl(Age, Salary)` ist das Ergebnis 0,9270611. Das Ergebnis wird nur für Zellen mit Gesamtwerten angezeigt.

Beispiel	Ergebnis
`Correl (TOTAL Age, Salary))`	0.927. Dieses und die folgenden Ergebnisse werden zur besseren Lesbarkeit mit drei Nachkommastellen angegeben. Wenn Sie ein Filterfenster mit der Dimension Gender erstellen und darin Auswahlen vornehmen, wird bei Auswahl von Female 0,951 und bei Auswahl von Male 0,939 ausgegeben. Dies geschieht, weil die Auswahl alle Ergebnisse ausschließt, die zum jeweils anderen Wert für Gender gehören.
`Correl({1} TOTAL Age, Salary))`	0.927. Unabhängig von Auswahlen. Dies geschieht, weil alle gewählten Optionen und Dimensionen von der Auswahlformel {1} nicht berücksichtigt werden.
`Correl (TOTAL <Gender> Age, Salary))`	0,927 in der Zelle mit Gesamtwert, 0,939 für alle Werte Male und 0,951 für alle Werte Female. Das entspricht den Ergebnissen durch Auswahlen im Filterfenster auf Grundlage von Gender.

In Beispielen verwendete Daten:

```
Salary:

LOAD * inline [

"Employee name"|Gender|Age|Salary

Aiden Charles|Male|20|25000

Brenda Davies|Male|25|32000

Charlotte Edberg|Female|45|56000

Daroush Ferrara|Male|31|29000

Eunice Goldblum|Female|31|32000

Freddy Halvorsen|Male|25|26000

Gauri Indu|Female|36|46000

Harry Jones|Male|38|40000

Ian Underwood|Male|40|45000

Jackie Kingsley|Female|23|28000

] (delimiter is '|');
```

Siehe auch:

Aggr - Diagrammfunktion (page 566)

- *Avg - Diagrammfunktion (page 419)*
- *RangeCorrel (page 1366)*

Fractile

Fractile() liefert den Wert, der dem inklusiven Fraktil (Quantil) der aggregierten Daten in der Formel über mehrere Datensätze entspricht, wie in einer **group by**-Klausel definiert wurde.

Sie können FractileExc (page 428) verwenden, um das exklusive Fraktil zu berechnen.

Syntax:

```
Fractile(expr, fraction)
```

Rückgabe Datentyp: numerisch

Die Funktion gibt den dem Rang entsprechenden Wert zurück, wie unter `rank = fraction * (N-1) + 1` definiert, wobei `N` die Anzahl der Werte in `expr` ist. Wenn `rank` eine Dezimalzahl ist, erfolgt eine Interpolation zwischen den zwei nächstliegenden Werten.

Argumente:

Argumente

Argument	Beschreibung
expr	Die Formel oder das Feld mit den Daten, die beim Berechnen des Fraktils verwendet werden sollen.
fraction	Eine Zahl zwischen 0 und 1, die dem zu berechnenden Fraktil (Quantil als Bruchteil ausgedrückt) entspricht.

Beispiele und Ergebnisse:

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Fügen Sie einem Arbeitsblatt in Ihrer App dann die Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um das Ergebnis anzuzeigen.

Ergebnisdaten

Beispiel	Ergebnis
Table1: Crosstable (Type, Value) Load recno() as ID, * inline [Observation|Comparison 35|2 40|27 12|38 15|31 21|1 14|19 46|1 10|34 28|3 48|1 16|2 30|3 32|2 48|1 31|2 22|1 12|3 39|29 19|37 25|2] (delimiter is '|'); Fractile1: LOAD Type, Fractile(Value,0.75) as MyFractile Resident Table1 Group By Type;	In einer Tabelle mit den Dimensionen Type und MyFractile werden die Ergebnisse der Berechnungen von Fractile() im Datenladeskript angezeigt: Type MyFractile Comparison 27.5 Observation 36

Fractile - Diagrammfunktion

Fractile() sucht nach dem Wert, der dem inklusiven Fraktil (Quantil) der aggregierten Daten in dem Bereich entspricht, der von der Formel über die Dimensionen des Diagramms vorgegeben ist.

Sie können FractileExc - Diagrammfunktion (page 430) verwenden, um das exklusive Fraktil zu berechnen.

Syntax:

```
Fractile([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr, fraction)
```

Rückgabe Datentyp: numerisch

Die Funktion gibt den dem Rang entsprechenden Wert zurück, wie unter `rank = fraction * (N-1) + 1` definiert, wobei `N` die Anzahl der Werte in `expr` ist. Wenn `rank` eine Dezimalzahl ist, erfolgt eine Interpolation zwischen den zwei nächstliegenden Werten.

Argumente:

Argumente

Argument	Beschreibung
expr	Die Formel oder das Feld mit den Daten, die beim Berechnen des Fraktils verwendet werden sollen.
fraction	Eine Zahl zwischen 0 und 1, die dem zu berechnenden Fraktil (Quantil als Bruchteil ausgedrückt) entspricht.
SetExpression	Standardmäßig berechnet sich die Aggregierungsfunktion über alle wählbaren Werte. Alternativ können Sie die der Berechnung zugrunde liegenden Werte über die Auswahlformel bestimmen.
DISTINCT	Der Zusatz **DISTINCT** vor den Funktionsargumenten bewirkt, dass bei der Auswertung der Funktionsargumente entstehende Duplikate nicht berücksichtigt werden.
TOTAL	Der Zusatz **TOTAL** vor der Funktion bewirkt, dass die Berechnung über alle ausgewählten bzw. wählbaren Werte erfolgt, und nicht nur über diejenigen, die zu dem Wert der aktuellen Dimension zählen. Die Dimensionen des Diagramms werden also nicht berücksichtigt. Mit **TOTAL [<fld {.fld}>]**, wobei auf den Zusatz **TOTAL** eine Liste aus mindestens einem Feldnamen (d. h. einer Teilmenge der Diagrammdimensionsvariablen) folgt, erstellen Sie eine Teilmenge aller möglichen Werte.

Beschränkungen:

Die Parameter der Aggregierungsfunktion dürfen keine anderen Aggregierungsfunktionen enthalten, es sei denn, diese inneren Aggregierungen enthalten den Qualifizierer **TOTAL**. Für komplexere verschachtelte Aggregierungen verwenden Sie die erweiterte Funktion **Aggr** in Verbindung mit einer angegebenen Dimension.

Beispiele und Ergebnisse:

Example table

Customer	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Astrida	46	60	70	13	78	20	45	65	78	12	78	22
Betacab	65	56	22	79	12	56	45	24	32	78	55	15
Canutility	77	68	34	91	24	68	57	36	44	90	67	27
Divadip	57	36	44	90	67	27	57	68	47	90	80	94

Funktionsbeispiele

Beispiel	Ergebnis
`Fractile (Sales, 0.75)`	Für eine Tabelle mit der Dimension `Customer` und der Kennzahl `Fractile([Sales])` ist das Ergebnis 71.75, wenn **Gesamtwerte** angezeigt werden. Hierbei handelt es sich um den Punkt in der Verteilung der Werte von `Sales`, unter dem 75 % aller Werte liegen.
`Fractile (TOTAL Sales, 0.75))`	71,75 für alle Werte von `Customer`, da durch den Zusatz TOTAL die Dimensionen nicht berücksichtigt werden.
`Fractile (DISTINCT Sales, 0.75)`	70 für den Gesamtwert, da durch den Zusatz DISTINCT in `Sales` für jeden `Customer` nur eindeutige Werte berechnet werden.

In Beispielen verwendete Daten:

```
Monthnames:
LOAD *, Dual(MonthText,MonthNumber) as Month INLINE [
MonthText, MonthNumber
Jan, 1
Feb, 2
Mar, 3
Apr, 4
May, 5
Jun, 6
Jul, 7
Aug, 8
Sep, 9
Oct, 10
Nov, 11
Dec, 12
];

Sales2013:
Crosstable (MonthText, Sales) LOAD * inline [
Customer|Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec
Astrida|46|60|70|13|78|20|45|65|78|12|78|22
Betacab|65|56|22|79|12|56|45|24|32|78|55|15
Canutility|77|68|34|91|24|68|57|36|44|90|67|27
Divadip|57|36|44|90|67|27|57|68|47|90|80|94
] (delimiter is '|');
```

Siehe auch:

Aggr - Diagrammfunktion (page 566)

FractileExc

FractileExc() liefert den Wert, der dem exklusiven Fraktil (Quantil) der aggregierten Daten in der Formel über mehrere Datensätze entspricht, wie in einer **group by**-Klausel definiert wurde.

Sie können Fractile (page 425) verwenden, um das inklusive Fraktil zu berechnen.

Syntax:

```
FractileExc(expr, fraction)
```

Rückgabe Datentyp: numerisch

Die Funktion gibt den dem Rang entsprechenden Wert zurück, wie unter `rank = fraction * (N+1)` definiert, wobei `N` die Anzahl der Werte in `expr` ist. Wenn `rank` eine Dezimalzahl ist, erfolgt eine Interpolation zwischen den zwei nächstliegenden Werten.

Argumente:

Argumente

Argument	Beschreibung
expr	Die Formel oder das Feld mit den Daten, die beim Berechnen des Fraktils verwendet werden sollen.
fraction	Eine Zahl zwischen 0 und 1, die dem zu berechnenden Fraktil (Quantil als Bruchteil ausgedrückt) entspricht.

Beispiele und Ergebnisse:

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Fügen Sie einem Arbeitsblatt in Ihrer App dann die Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um das Ergebnis anzuzeigen.

Ergebnisdaten

Beispiel	Ergebnis
Table1: Crosstable (Type, Value) Load recno() as ID, * inline [Observation|Comparison 35|2 40|27 12|38 15|31 21|1 14|19 46|1 10|34 28|3 48|1 16|2 30|3 32|2 48|1 31|2 22|1 12|3 39|29 19|37 25|2] (delimiter is '|'); Fractile1: LOAD Type, FractileExc(Value,0.75) as MyFractile Resident Table1 Group By Type;	In einer Tabelle mit den Dimensionen Type und MyFractile werden die Ergebnisse der Berechnungen von FractileExc() im Datenladeskript angezeigt: Type MyFractile Comparison 28.5 Observation 38

FractileExc - Diagrammfunktion

FractileExc() sucht nach dem Wert, der dem exklusiven Fraktil (Quantil) der aggregierten Daten in dem Bereich entspricht, der von der Formel über die Dimensionen des Diagramms vorgegeben ist.

Sie können Fractile - Diagrammfunktion (page 426) verwenden, um das inklusive Fraktil zu berechnen.

Syntax:

```
FractileExc([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr,
fraction)
```

Rückgabe Datentyp: numerisch

Die Funktion gibt den dem Rang entsprechenden Wert zurück, wie unter `rank = fraction * (N+1)` definiert, wobei `N` die Anzahl der Werte in `expr` ist. Wenn `rank` eine Dezimalzahl ist, erfolgt eine Interpolation zwischen den zwei nächstliegenden Werten.

Argumente:

Argumente

Argument	Beschreibung
expr	Die Formel oder das Feld mit den Daten, die beim Berechnen des Fraktils verwendet werden sollen.
fraction	Eine Zahl zwischen 0 und 1, die dem zu berechnenden Fraktil (Quantil als Bruchteil ausgedrückt) entspricht.
SetExpression	Standardmäßig berechnet sich die Aggregierungsfunktion über alle wählbaren Werte. Alternativ können Sie die der Berechnung zugrunde liegenden Werte über die Auswahlformel bestimmen.
DISTINCT	Der Zusatz **DISTINCT** vor den Funktionsargumenten bewirkt, dass bei der Auswertung der Funktionsargumente entstehende Duplikate nicht berücksichtigt werden.
TOTAL	Der Zusatz **TOTAL** vor der Funktion bewirkt, dass die Berechnung über alle ausgewählten bzw. wählbaren Werte erfolgt, und nicht nur über diejenigen, die zu dem Wert der aktuellen Dimension zählen. Die Dimensionen des Diagramms werden also nicht berücksichtigt. Mit **TOTAL [<fld {.fld}>]**, wobei auf den Zusatz **TOTAL** eine Liste aus mindestens einem Feldnamen (d. h. einer Teilmenge der Diagrammdimensionsvariablen) folgt, erstellen Sie eine Teilmenge aller möglichen Werte.

Beschränkungen:

Die Parameter der Aggregierungsfunktion dürfen keine anderen Aggregierungsfunktionen enthalten, es sei denn, diese inneren Aggregierungen enthalten den Qualifizierer **TOTAL**. Für komplexere verschachtelte Aggregierungen verwenden Sie die erweiterte Funktion **Aggr** in Verbindung mit einer angegebenen Dimension.

Beispiele und Ergebnisse:

Example table

Customer	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Astrida	46	60	70	13	78	20	45	65	78	12	78	22
Betacab	65	56	22	79	12	56	45	24	32	78	55	15
Canutility	77	68	34	91	24	68	57	36	44	90	67	27
Divadip	57	36	44	90	67	27	57	68	47	90	80	94

Funktionsbeispiele

Beispiel	Ergebnis
`FractileExc (Sales, 0.75)`	Für eine Tabelle mit der Dimension `Customer` und der Kennzahl `FractileExc([Sales])` ist das Ergebnis 75.25, wenn **Gesamtwerte** angezeigt werden. Hierbei handelt es sich um den Punkt in der Verteilung der Werte von `Sales`, unter dem 75 % aller Werte liegen.
`FractileExc (TOTAL Sales, 0.75))`	71,25 für alle Werte von `Customer`, da durch den Qualifizierer TOTAL die Dimensionen nicht berücksichtigt werden.
`FractileExc (DISTINCT Sales, 0.75)`	73,50 für den Gesamtwert, da durch den Qualifizierer DISTINCT in `Sales` für jeden `Customer` nur eindeutige Werte berechnet werden.

In Beispielen verwendete Daten:

```
Monthnames:
LOAD *, Dual(MonthText,MonthNumber) as Month INLINE [
MonthText, MonthNumber
Jan, 1
Feb, 2
Mar, 3
Apr, 4
May, 5
Jun, 6
Jul, 7
Aug, 8
Sep, 9
Oct, 10
Nov, 11
Dec, 12
];

Sales2013:
Crosstable (MonthText, Sales) LOAD * inline [
Customer|Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec
Astrida|46|60|70|13|78|20|45|65|78|12|78|22
Betacab|65|56|22|79|12|56|45|24|32|78|55|15
Canutility|77|68|34|91|24|68|57|36|44|90|67|27
Divadip|57|36|44|90|67|27|57|68|47|90|80|94
] (delimiter is '|');
```

Siehe auch:

Aggr - Diagrammfunktion (page 566)

Kurtosis

Kurtosis() liefert die Kurtosis der Daten in der Formel über mehrere Datensätze, wie in einer Bedingung **group by** definiert wurde.

Syntax:

```
Kurtosis([distinct ] expr )
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
expr	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.
distinct	Ist der Formel das Wort **distinct** vorangestellt, werden Dubletten nicht berücksichtigt.

Beispiele und Ergebnisse:

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Fügen Sie einem Arbeitsblatt in Ihrer App dann die Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um das Ergebnis anzuzeigen.

Ergebnisdaten

Beispiel	Ergebnis
Table1: Crosstable (Type, Value) Load recno() as ID, * inline [Observation|Comparison 35|2 40|27 12|38 15|31 21|1 14|19 46|1 10|34 28|3 48|1 16|2 30|3 32|2 48|1 31|2 22|1 12|3 39|29 19|37 25|2] (delimiter is '|'); Kurtosis1: LOAD Type, Kurtosis(Value) as MyKurtosis1, Kurtosis(DISTINCT Value) as MyKurtosis2 Resident Table1 Group By Type;	In einer Tabelle mit den Dimensionen Type, MyKurtosis1 und MyKurtosis2 werden die Ergebnisse der Berechnungen von Kurtosis() im Datenladeskript angezeigt: Type MyKurtosis1 MyKurtosis2 Comparison -1.1612957 -1.4982366 Observation -1.1148768 -0.93540144

Kurtosis - Diagrammfunktion

Kurtosis() sucht nach der Kurtosis des in der Formel oder im Feld aggregierten Bereichs von Daten, die über die Dimensionen des Diagramms iteriert wurden.

Syntax:

```
Kurtosis([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr)
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
expr	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.

Argument	Beschreibung
SetExpression	Standardmäßig berechnet sich die Aggregierungsfunktion über alle wählbaren Werte. Alternativ können Sie die der Berechnung zugrunde liegenden Werte über die Auswahlformel bestimmen.
DISTINCT	Der Zusatz **DISTINCT** vor den Funktionsargumenten bewirkt, dass bei der Auswertung der Funktionsargumente entstehende Duplikate nicht berücksichtigt werden.
TOTAL	Der Zusatz **TOTAL** vor der Funktion bewirkt, dass die Berechnung über alle ausgewählten bzw. wählbaren Werte erfolgt, und nicht nur über diejenigen, die zu dem Wert der aktuellen Dimension zählen. Die Dimensionen des Diagramms werden also nicht berücksichtigt. Mit **TOTAL [<fld {.fld}>]**, wobei auf den Zusatz **TOTAL** eine Liste aus mindestens einem Feldnamen (d. h. einer Teilmenge der Diagrammdimensionsvariablen) folgt, erstellen Sie eine Teilmenge aller möglichen Werte.

Beschränkungen:

Die Parameter der Aggregierungsfunktion dürfen keine anderen Aggregierungsfunktionen enthalten, es sei denn, diese inneren Aggregierungen enthalten den Qualifizierer **TOTAL**. Für komplexere verschachtelte Aggregierungen verwenden Sie die erweiterte Funktion **Aggr** in Verbindung mit einer angegebenen Dimension.

Beispiele und Ergebnisse:

Example table

Type	Value																			
Comparison	2	27	38	31	1	19	1	34	3	1	2	3	2	1	2	1	3	29	37	2
Observation	35	40	12	15	21	14	46	10	28	48	16	30	32	48	31	22	12	39	19	25

Funktionsbeispiele

Beispiel	Ergebnis
`Kurtosis (Value)`	Für eine Tabelle mit der Dimension `Type` und der Kennzahl `Kurtosis(Value)` ist das Ergebnis 1,252, wenn **Gesamtwerte** für die Tabelle angezeigt werden und die Zahlenformatierung auf 3 Dezimalstellen beschränkt ist. Für `Comparison` beträgt der Wert 1,161 und für `Observation` 1,115.
`Kurtosis (TOTAL Value))`	1,252 für alle Werte von `Type`, da durch den Zusatz TOTAL die Dimensionen nicht berücksichtigt werden.

In Beispielen verwendete Daten:

```
Table1:
Crosstable (Type, Value)
Load recno() as ID, * inline [
Observation|Comparison
35|2
40|27
12|38
15|31
21|1
14|19
46|1
10|34
28|3
48|1
16|2
30|3
32|2
48|1
31|2
22|1
12|3
39|29
19|37
25|2 ] (delimiter is '|');
```

Siehe auch:

Avg - Diagrammfunktion (page 419)

LINEST_B

LINEST_B() liefert den y-Achsenabschnitt (Regressionskonstante) b in der durch die Gleichung y=mx+b bestimmten linearen Regression für eine Reihe von Koordinaten, die durch x-expression und y-expression definiert sind, über die im **group by**-Zusatz bezeichneten Datensätze.

Syntax:

```
LINEST_B (y_value, x_value[, y0 [, x0 ]])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
y_value	Die Formel oder das Feld mit dem Datenbereich y, der angegeben werden soll.
x_value	Die Formel oder das Feld mit dem Datenbereich x, der angegeben werden soll.

Argument	Beschreibung
y(0), x(0)	Der optionale Parameter y0definiert, bei welchem Wert die Regressionsgerade die y-Achse schneidet. Die optionalen Parameter y0 und x0 definieren einen Koordinatenpunkt, durch den die Regressionsgerade geht. Sofern die Parameter y0 und x0 nicht definiert sind, erfordert die Berechnung zwei gültige Datenpaare. Wenn y0 und x0 definiert sind, reicht ein Datenpaar.

Beschränkungen:

Textwerte, NULL-Werte und fehlende Werte in einem oder beiden Teilen eines Wertepaars führen dazu, dass das Wertepaar ignoriert wird.

Siehe auch:

Beispiele zur Verwendung von linest-Funktionen (page 478)

LINEST_B - Diagrammfunktion

LINEST_B() liefert den Wert b (y-Achsenabschnitt) in der durch die Gleichung y=mx+b bestimmten linearen Regression für eine Reihe von Koordinaten, die durch **x_value** und **y_value** definiert sind, aggregiert nach den Dimensionen des Diagramms.

Syntax:

```
LINEST_B([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value
[, y0_const [ , x0_const]])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
y_value	Die Formel oder das Feld mit dem Datenbereich y, der angegeben werden soll.
x_value	Die Formel oder das Feld mit dem Datenbereich x, der angegeben werden soll.
y0_const, x0_const	Der optionale Parameter y0definiert, bei welchem Wert die Regressionsgerade die y-Achse schneidet. Die optionalen Parameter y0 und x0 definieren einen Koordinatenpunkt, durch den die Regressionsgerade geht. *Sofern die Parameter y0 und x0 nicht definiert sind, erfordert die Berechnung zwei gültige Datenpaare. Wenn y0 und x0 definiert sind, reicht ein Datenpaar.*

Argument	Beschreibung
SetExpression	Standardmäßig berechnet sich die Aggregierungsfunktion über alle wählbaren Werte. Alternativ können Sie die der Berechnung zugrunde liegenden Werte über die Auswahlformel bestimmen.
DISTINCT	Der Zusatz **DISTINCT** vor den Funktionsargumenten bewirkt, dass bei der Auswertung der Funktionsargumente entstehende Duplikate nicht berücksichtigt werden.
TOTAL	Der Zusatz **TOTAL** vor der Funktion bewirkt, dass die Berechnung über alle ausgewählten bzw. wählbaren Werte erfolgt, und nicht nur über diejenigen, die zu dem Wert der aktuellen Dimension zählen. Die Dimensionen des Diagramms werden also nicht berücksichtigt. Mit **TOTAL [<fld {.fld}>]**, wobei auf den Zusatz **TOTAL** eine Liste aus mindestens einem Feldnamen (d. h. einer Teilmenge der Diagrammdimensionsvariablen) folgt, erstellen Sie eine Teilmenge aller möglichen Werte.

Beschränkungen:

Die Parameter der Aggregierungsfunktion dürfen keine anderen Aggregierungsfunktionen enthalten, es sei denn, diese inneren Aggregierungen enthalten den Qualifizierer **TOTAL**. Für komplexere verschachtelte Aggregierungen verwenden Sie die erweiterte Funktion **Aggr** in Verbindung mit einer angegebenen Dimension.

Textwerte, NULL-Werte und fehlende Werte in einem oder beiden Teilen eines Wertepaars führen dazu, dass das Wertepaar ignoriert wird.

Siehe auch:

Beispiele zur Verwendung von linest-Funktionen (page 478)
Avg - Diagrammfunktion (page 419)

LINEST_DF

LINEST_DF() liefert die aggregierten Freiheitsgrade in der durch die Gleichung y=mx+b bestimmten linearen Regression für eine Reihe von Koordinaten, die durch x-expression und y-expression definiert sind, aggregiert über die in einer **group by**-Bedingung bezeichneten Datensätze.

Syntax:

```
LINEST_DF (y_value, x_value[, y0 [, x0 ]])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
y_value	Die Formel oder das Feld mit dem Datenbereich y, der angegeben werden soll.
x_value	Die Formel oder das Feld mit dem Datenbereich x, der angegeben werden soll.
y(0), x(0)	Der optionale Parameter y0definiert, bei welchem Wert die Regressionsgerade die y-Achse schneidet. Die optionalen Parameter y0 und x0 definieren einen Koordinatenpunkt, durch den die Regressionsgerade geht. Sofern die Parameter y0 und x0 nicht definiert sind, erfordert die Berechnung zwei gültige Datenpaare. Wenn y0 und x0 definiert sind, reicht ein Datenpaar.

Beschränkungen:

Textwerte, NULL-Werte und fehlende Werte in einem oder beiden Teilen eines Wertepaars führen dazu, dass das Wertepaar ignoriert wird.

Siehe auch:

Beispiele zur Verwendung von linest-Funktionen (page 478)

LINEST_DF - Diagrammfunktion

LINEST_DF() liefert die Freiheitsgrade der durch die Gleichung y=mx+b bestimmten linearen Regression für eine Reihe von Koordinaten, die durch **x_value** und **y_value** definiert sind, aggregiert nach den Dimensionen des Diagramms.

Syntax:

```
LINEST_DF([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_
value [, y0_const [, x0_const]])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
y_value	Die Formel oder das Feld mit dem Datenbereich y, der angegeben werden soll.
x_value	Die Formel oder das Feld mit dem Datenbereich x, der angegeben werden soll.

Argument	Beschreibung
y0, x0	Der optionale Parameter y0definiert, bei welchem Wert die Regressionsgerade die y-Achse schneidet. Die optionalen Parameter y0 und x0 definieren einen Koordinatenpunkt, durch den die Regressionsgerade geht. *Sofern die Parameter y0 und x0 nicht definiert sind, erfordert die Berechnung zwei gültige Datenpaare. Wenn y0 und x0 definiert sind, reicht ein Datenpaar.*
SetExpression	Standardmäßig berechnet sich die Aggregierungsfunktion über alle wählbaren Werte. Alternativ können Sie die der Berechnung zugrunde liegenden Werte über die Auswahlformel bestimmen.
DISTINCT	Der Zusatz **DISTINCT** vor den Funktionsargumenten bewirkt, dass bei der Auswertung der Funktionsargumente entstehende Duplikate nicht berücksichtigt werden.
TOTAL	Der Zusatz **TOTAL** vor der Funktion bewirkt, dass die Berechnung über alle ausgewählten bzw. wählbaren Werte erfolgt, und nicht nur über diejenigen, die zu dem Wert der aktuellen Dimension zählen. Die Dimensionen des Diagramms werden also nicht berücksichtigt. Mit **TOTAL [<fld {.fld}>]**, wobei auf den Zusatz **TOTAL** eine Liste aus mindestens einem Feldnamen (d. h. einer Teilmenge der Diagrammdimensionsvariablen) folgt, erstellen Sie eine Teilmenge aller möglichen Werte.

Beschränkungen:

Die Parameter der Aggregierungsfunktion dürfen keine anderen Aggregierungsfunktionen enthalten, es sei denn, diese inneren Aggregierungen enthalten den Qualifizierer **TOTAL**. Für komplexere verschachtelte Aggregierungen verwenden Sie die erweiterte Funktion **Aggr** in Verbindung mit einer angegebenen Dimension.

Textwerte, NULL-Werte und fehlende Werte in einem oder beiden Teilen eines Wertepaars führen dazu, dass das Wertepaar ignoriert wird.

Siehe auch:

Beispiele zur Verwendung von linest-Funktionen (page 478)
Avg - Diagrammfunktion (page 419)

LINEST_F

Diese Skriptfunktion liefert den F-Wert ($r^2/(1-r^2)$), der durch die Gleichung y=mx+b bestimmten linearen Regression für eine Reihe von Koordinaten, die durch x-expression und y-expression definiert sind, aggregiert über eine durch den **group by**-Zusatz festgelegte Anzahl von Datensätzen.

Syntax:

```
LINEST_F (y_value, x_value[, y0 [, x0 ]])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
y_value	Die Formel oder das Feld mit dem Datenbereich y, der angegeben werden soll.
x_value	Die Formel oder das Feld mit dem Datenbereich x, der angegeben werden soll.
y(0), x(0)	Der optionale Parameter y0definiert, bei welchem Wert die Regressionsgerade die y-Achse schneidet. Die optionalen Parameter y0 und x0 definieren einen Koordinatenpunkt, durch den die Regressionsgerade geht. Sofern die Parameter y0 und x0 nicht definiert sind, erfordert die Berechnung zwei gültige Datenpaare. Wenn y0 und x0 definiert sind, reicht ein Datenpaar.

Beschränkungen:

Textwerte, NULL-Werte und fehlende Werte in einem oder beiden Teilen eines Wertepaars führen dazu, dass das Wertepaar ignoriert wird.

Siehe auch:

Beispiele zur Verwendung von linest-Funktionen (page 478)

LINEST_F - Diagrammfunktion

LINEST_F() liefert den F-Wert (r2/(1-r2)) der durch die Gleichung y=mx+b bestimmten linearen Regression für eine Reihe von Koordinaten, die durch **x_value** und **y_value** definiert sind, aggregiert nach den Dimensionen des Diagramms.

Syntax:

```
LINEST_F([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value
[, y0_const [, x0_const]])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
y_value	Die Formel oder das Feld mit dem Datenbereich y, der angegeben werden soll.
x_value	Die Formel oder das Feld mit dem Datenbereich x, der angegeben werden soll.

Argument	Beschreibung
y0, x0	Der optionale Parameter y0definiert, bei welchem Wert die Regressionsgerade die y-Achse schneidet. Die optionalen Parameter y0 und x0 definieren einen Koordinatenpunkt, durch den die Regressionsgerade geht. *Sofern die Parameter y0 und x0 nicht definiert sind, erfordert die Berechnung zwei gültige Datenpaare. Wenn y0 und x0 definiert sind, reicht ein Datenpaar.*
SetExpression	Standardmäßig berechnet sich die Aggregierungsfunktion über alle wählbaren Werte. Alternativ können Sie die der Berechnung zugrunde liegenden Werte über die Auswahlformel bestimmen.
DISTINCT	Der Zusatz **DISTINCT** vor den Funktionsargumenten bewirkt, dass bei der Auswertung der Funktionsargumente entstehende Duplikate nicht berücksichtigt werden.
TOTAL	Der Zusatz **TOTAL** vor der Funktion bewirkt, dass die Berechnung über alle ausgewählten bzw. wählbaren Werte erfolgt, und nicht nur über diejenigen, die zu dem Wert der aktuellen Dimension zählen. Die Dimensionen des Diagramms werden also nicht berücksichtigt. Mit **TOTAL [<fld {.fld}>]**, wobei auf den Zusatz **TOTAL** eine Liste aus mindestens einem Feldnamen (d. h. einer Teilmenge der Diagrammdimensionsvariablen) folgt, erstellen Sie eine Teilmenge aller möglichen Werte.

Beschränkungen:

Die Parameter der Aggregierungsfunktion dürfen keine anderen Aggregierungsfunktionen enthalten, es sei denn, diese inneren Aggregierungen enthalten den Qualifizierer **TOTAL**. Für komplexere verschachtelte Aggregierungen verwenden Sie die erweiterte Funktion **Aggr** in Verbindung mit einer angegebenen Dimension.

Textwerte, NULL-Werte und fehlende Werte in einem oder beiden Teilen eines Wertepaars führen dazu, dass das Wertepaar ignoriert wird.

Siehe auch:

Beispiele zur Verwendung von linest-Funktionen (page 478)
Avg - Diagrammfunktion (page 419)

LINEST_M

LINEST_M() liefert den m-Wert (Steigung) in der durch die Gleichung y=mx+b bestimmten linearen Regression für eine Reihe von Koordinaten, die durch x-expression und y-expression definiert sind, aggregiert über die in einer **group by**-Bedingung bezeichneten Datensätze.

Syntax:

```
LINEST_M (y_value, x_value[, y0 [, x0 ]])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
y_value	Die Formel oder das Feld mit dem Datenbereich y, der angegeben werden soll.
x_value	Die Formel oder das Feld mit dem Datenbereich x, der angegeben werden soll.
y(0), x(0)	Der optionale Parameter y0definiert, bei welchem Wert die Regressionsgerade die y-Achse schneidet. Die optionalen Parameter y0 und x0 definieren einen Koordinatenpunkt, durch den die Regressionsgerade geht. Sofern die Parameter y0 und x0 nicht definiert sind, erfordert die Berechnung zwei gültige Datenpaare. Wenn y0 und x0 definiert sind, reicht ein Datenpaar.

Beschränkungen:

Textwerte, NULL-Werte und fehlende Werte in einem oder beiden Teilen eines Wertepaars führen dazu, dass das Wertepaar ignoriert wird.

Siehe auch:

Beispiele zur Verwendung von linest-Funktionen (page 478)

LINEST_M - Diagrammfunktion

LINEST_M() liefert den m-Wert (Steigung) in der durch die Gleichung y=mx+b bestimmten linearen Regression für eine Reihe von Koordinaten, die durch **x_value** und **y_value** definiert sind, aggregiert nach den Dimensionen des Diagramms.

Syntax:

```
LINEST_M([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value
[, y0_const [, x0_const]])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
y_value	Die Formel oder das Feld mit dem Datenbereich y, der angegeben werden soll.
x_value	Die Formel oder das Feld mit dem Datenbereich x, der angegeben werden soll.

Argument	Beschreibung
y0, x0	Der optionale Parameter y0definiert, bei welchem Wert die Regressionsgerade die y-Achse schneidet. Die optionalen Parameter y0 und x0 definieren einen Koordinatenpunkt, durch den die Regressionsgerade geht. *Sofern die Parameter y0 und x0 nicht definiert sind, erfordert die Berechnung zwei gültige Datenpaare. Wenn y0 und x0 definiert sind, reicht ein Datenpaar.*
SetExpression	Standardmäßig berechnet sich die Aggregierungsfunktion über alle wählbaren Werte. Alternativ können Sie die der Berechnung zugrunde liegenden Werte über die Auswahlformel bestimmen.
DISTINCT	Der Zusatz **DISTINCT** vor den Funktionsargumenten bewirkt, dass bei der Auswertung der Funktionsargumente entstehende Duplikate nicht berücksichtigt werden.
TOTAL	Der Zusatz **TOTAL** vor der Funktion bewirkt, dass die Berechnung über alle ausgewählten bzw. wählbaren Werte erfolgt, und nicht nur über diejenigen, die zu dem Wert der aktuellen Dimension zählen. Die Dimensionen des Diagramms werden also nicht berücksichtigt. Mit **TOTAL [<fld {.fld}>]**, wobei auf den Zusatz **TOTAL** eine Liste aus mindestens einem Feldnamen (d. h. einer Teilmenge der Diagrammdimensionsvariablen) folgt, erstellen Sie eine Teilmenge aller möglichen Werte.

Beschränkungen:

Die Parameter der Aggregierungsfunktion dürfen keine anderen Aggregierungsfunktionen enthalten, es sei denn, diese inneren Aggregierungen enthalten den Qualifizierer **TOTAL**. Für komplexere verschachtelte Aggregierungen verwenden Sie die erweiterte Funktion **Aggr** in Verbindung mit einer angegebenen Dimension.

Textwerte, NULL-Werte und fehlende Werte in einem oder beiden Teilen eines Wertepaars führen dazu, dass das Wertepaar ignoriert wird.

Siehe auch:

Beispiele zur Verwendung von linest-Funktionen (page 478)
Avg - Diagrammfunktion (page 419)

LINEST_R2

LINEST_R2() liefert den aggregierten Wert r^2 (Bestimmtheitsmaß) einer durch die Gleichung y=mx+b bestimmten linearen Regression für eine Reihe von Koordinaten, die durch x-expression und y-expression definiert sind, aggregiert über eine Anzahl von Datensätzen, die durch eine **group by**-Bedingung definiert sind.

Syntax:

```
LINEST_R2 (y_value, x_value[, y0 [, x0 ]])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
y_value	Die Formel oder das Feld mit dem Datenbereich y, der angegeben werden soll.
x_value	Die Formel oder das Feld mit dem Datenbereich x, der angegeben werden soll.
y(0), x(0)	Der optionale Parameter y0definiert, bei welchem Wert die Regressionsgerade die y-Achse schneidet. Die optionalen Parameter y0 und x0 definieren einen Koordinatenpunkt, durch den die Regressionsgerade geht. Sofern die Parameter y0 und x0 nicht definiert sind, erfordert die Berechnung zwei gültige Datenpaare. Wenn y0 und x0 definiert sind, reicht ein Datenpaar.

Beschränkungen:

Textwerte, NULL-Werte und fehlende Werte in einem oder beiden Teilen eines Wertepaars führen dazu, dass das Wertepaar ignoriert wird.

Siehe auch:

Beispiele zur Verwendung von linest-Funktionen (page 478)

LINEST_R2 - Diagrammfunktion

LINEST_R2() liefert den r2-Wert (Determinationskoeffizient) in der durch die Gleichung y=mx+b bestimmten linearen Regression für eine Reihe von Koordinaten, die durch **x_value** und **y_value** definiert sind, aggregiert nach den Dimensionen des Diagramms.

Syntax:

```
LINEST_R2([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_
value[, y0_const[, x0_const]])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
y_value	Die Formel oder das Feld mit dem Datenbereich y, der angegeben werden soll.
x_value	Die Formel oder das Feld mit dem Datenbereich x, der angegeben werden soll.

Argument	Beschreibung
y0, x0	Der optionale Parameter y0definiert, bei welchem Wert die Regressionsgerade die y-Achse schneidet. Die optionalen Parameter y0 und x0 definieren einen Koordinatenpunkt, durch den die Regressionsgerade geht. *Sofern die Parameter y0 und x0 nicht definiert sind, erfordert die Berechnung zwei gültige Datenpaare. Wenn y0 und x0 definiert sind, reicht ein Datenpaar.*
SetExpression	Standardmäßig berechnet sich die Aggregierungsfunktion über alle wählbaren Werte. Alternativ können Sie die der Berechnung zugrunde liegenden Werte über die Auswahlformel bestimmen.
DISTINCT	Der Zusatz **DISTINCT** vor den Funktionsargumenten bewirkt, dass bei der Auswertung der Funktionsargumente entstehende Duplikate nicht berücksichtigt werden.
TOTAL	Der Zusatz **TOTAL** vor der Funktion bewirkt, dass die Berechnung über alle ausgewählten bzw. wählbaren Werte erfolgt, und nicht nur über diejenigen, die zu dem Wert der aktuellen Dimension zählen. Die Dimensionen des Diagramms werden also nicht berücksichtigt. Mit **TOTAL [<fld {.fld}>]**, wobei auf den Zusatz **TOTAL** eine Liste aus mindestens einem Feldnamen (d. h. einer Teilmenge der Diagrammdimensionsvariablen) folgt, erstellen Sie eine Teilmenge aller möglichen Werte.

Beschränkungen:

Die Parameter der Aggregierungsfunktion dürfen keine anderen Aggregierungsfunktionen enthalten, es sei denn, diese inneren Aggregierungen enthalten den Qualifizierer **TOTAL**. Für komplexere verschachtelte Aggregierungen verwenden Sie die erweiterte Funktion **Aggr** in Verbindung mit einer angegebenen Dimension.

Textwerte, NULL-Werte und fehlende Werte in einem oder beiden Teilen eines Wertepaars führen dazu, dass das Wertepaar ignoriert wird.

Siehe auch:

Beispiele zur Verwendung von linest-Funktionen (page 478)
Avg - Diagrammfunktion (page 419)

LINEST_SEB

LINEST_SEB() liefert den Standardfehler des b-Werts in der durch die Gleichung y=mx+b bestimmten linearen Regression für eine Reihe von Koordinaten, die durch x-expression und y-expression definiert sind, über die im **group by**-Zusatz bezeichneten Datensätze.

Syntax:

```
LINEST_SEB (y_value, x_value[, y0 [, x0 ]])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
y_value	Die Formel oder das Feld mit dem Datenbereich y, der angegeben werden soll.
x_value	Die Formel oder das Feld mit dem Datenbereich x, der angegeben werden soll.
y(0), x(0)	Der optionale Parameter y0definiert, bei welchem Wert die Regressionsgerade die y-Achse schneidet. Die optionalen Parameter y0 und x0 definieren einen Koordinatenpunkt, durch den die Regressionsgerade geht. Sofern die Parameter y0 und x0 nicht definiert sind, erfordert die Berechnung zwei gültige Datenpaare. Wenn y0 und x0 definiert sind, reicht ein Datenpaar.

Beschränkungen:

Textwerte, NULL-Werte und fehlende Werte in einem oder beiden Teilen eines Wertepaars führen dazu, dass das Wertepaar ignoriert wird.

Siehe auch:

Beispiele zur Verwendung von linest-Funktionen (page 478)

LINEST_SEB - Diagrammfunktion

LINEST_SEB() liefert den Standardfehler des b-Werts in der durch die Gleichung y=mx+b bestimmten linearen Regression für eine Reihe von Koordinaten, die durch **x_value** und **y_value** definiert sind, aggregiert nach den Dimensionen des Diagramms.

Syntax:

```
LINEST_SEB([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_
value[, y0_const[, x0_const]])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
y_value	Die Formel oder das Feld mit dem Datenbereich y, der angegeben werden soll.
x_value	Die Formel oder das Feld mit dem Datenbereich x, der angegeben werden soll.

Argument	Beschreibung
y0, x0	Der optionale Parameter y0definiert, bei welchem Wert die Regressionsgerade die y-Achse schneidet. Die optionalen Parameter y0 und x0 definieren einen Koordinatenpunkt, durch den die Regressionsgerade geht. *Sofern die Parameter y0 und x0 nicht definiert sind, erfordert die Berechnung zwei gültige Datenpaare. Wenn y0 und x0 definiert sind, reicht ein Datenpaar.*
SetExpression	Standardmäßig berechnet sich die Aggregierungsfunktion über alle wählbaren Werte. Alternativ können Sie die der Berechnung zugrunde liegenden Werte über die Auswahlformel bestimmen.
DISTINCT	Der Zusatz **DISTINCT** vor den Funktionsargumenten bewirkt, dass bei der Auswertung der Funktionsargumente entstehende Duplikate nicht berücksichtigt werden.
TOTAL	Der Zusatz **TOTAL** vor der Funktion bewirkt, dass die Berechnung über alle ausgewählten bzw. wählbaren Werte erfolgt, und nicht nur über diejenigen, die zu dem Wert der aktuellen Dimension zählen. Die Dimensionen des Diagramms werden also nicht berücksichtigt. Mit **TOTAL [<fld {.fld}>]**, wobei auf den Zusatz **TOTAL** eine Liste aus mindestens einem Feldnamen (d. h. einer Teilmenge der Diagrammdimensionsvariablen) folgt, erstellen Sie eine Teilmenge aller möglichen Werte.

Beschränkungen:

Die Parameter der Aggregierungsfunktion dürfen keine anderen Aggregierungsfunktionen enthalten, es sei denn, diese inneren Aggregierungen enthalten den Qualifizierer **TOTAL**. Für komplexere verschachtelte Aggregierungen verwenden Sie die erweiterte Funktion **Aggr** in Verbindung mit einer angegebenen Dimension.

Textwerte, NULL-Werte und fehlende Werte in einem oder beiden Teilen eines Wertepaars führen dazu, dass das Wertepaar ignoriert wird.

Siehe auch:

Beispiele zur Verwendung von linest-Funktionen (page 478)
Avg - Diagrammfunktion (page 419)

LINEST_SEM

LINEST_SEM() liefert den Standardfehler des m-Werts in der durch die Gleichung y=mx+b bestimmten linearen Regression für eine Reihe von Koordinaten, die durch x-expression und y-expression definiert sind, über die im **group by**-Zusatz bezeichneten Datensätze.

Syntax:

```
LINEST_SEM (y_value, x_value[, y0 [, x0 ]])
```

Rückgabe Datentyp: numerisch

Argumente:

Argument	Beschreibung
y_value	Die Formel oder das Feld mit dem Datenbereich y, der angegeben werden soll.
x_value	Die Formel oder das Feld mit dem Datenbereich x, der angegeben werden soll.
y(0), x(0)	Der optionale Parameter y0definiert, bei welchem Wert die Regressionsgerade die y-Achse schneidet. Die optionalen Parameter y0 und x0 definieren einen Koordinatenpunkt, durch den die Regressionsgerade geht. Sofern die Parameter y0 und x0 nicht definiert sind, erfordert die Berechnung zwei gültige Datenpaare. Wenn y0 und x0 definiert sind, reicht ein Datenpaar.

Beschränkungen:

Textwerte, NULL-Werte und fehlende Werte in einem oder beiden Teilen eines Wertepaars führen dazu, dass das Wertepaar ignoriert wird.

Siehe auch:

Beispiele zur Verwendung von linest-Funktionen (page 478)

LINEST_SEM - Diagrammfunktion

LINEST_SEM() liefert den Standardfehler des m-Werts in der durch die Gleichung y=mx+b bestimmten linearen Regression für eine Reihe von Koordinaten, die durch **x_value** und **y_value** definiert sind, aggregiert nach den Dimensionen des Diagramms.

Syntax:

```
LINEST_SEM([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value[, y0_const[, x0_const]])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
y_value	Die Formel oder das Feld mit dem Datenbereich y, der angegeben werden soll.
x_value	Die Formel oder das Feld mit dem Datenbereich x, der angegeben werden soll.

Argument	Beschreibung
y0, x0	Der optionale Parameter y0definiert, bei welchem Wert die Regressionsgerade die y-Achse schneidet. Die optionalen Parameter y0 und x0 definieren einen Koordinatenpunkt, durch den die Regressionsgerade geht. *Sofern die Parameter y0 und x0 nicht definiert sind, erfordert die Berechnung zwei gültige Datenpaare. Wenn y0 und x0 definiert sind, reicht ein Datenpaar.*
SetExpression	Standardmäßig berechnet sich die Aggregierungsfunktion über alle wählbaren Werte. Alternativ können Sie die der Berechnung zugrunde liegenden Werte über die Auswahlformel bestimmen.
DISTINCT	Der Zusatz **DISTINCT** vor den Funktionsargumenten bewirkt, dass bei der Auswertung der Funktionsargumente entstehende Duplikate nicht berücksichtigt werden.
TOTAL	Der Zusatz **TOTAL** vor der Funktion bewirkt, dass die Berechnung über alle ausgewählten bzw. wählbaren Werte erfolgt, und nicht nur über diejenigen, die zu dem Wert der aktuellen Dimension zählen. Die Dimensionen des Diagramms werden also nicht berücksichtigt. Mit **TOTAL [<fld {.fld}>]**, wobei auf den Zusatz **TOTAL** eine Liste aus mindestens einem Feldnamen (d. h. einer Teilmenge der Diagrammdimensionsvariablen) folgt, erstellen Sie eine Teilmenge aller möglichen Werte.

Beschränkungen:

Die Parameter der Aggregierungsfunktion dürfen keine anderen Aggregierungsfunktionen enthalten, es sei denn, diese inneren Aggregierungen enthalten den Qualifizierer **TOTAL**. Für komplexere verschachtelte Aggregierungen verwenden Sie die erweiterte Funktion **Aggr** in Verbindung mit einer angegebenen Dimension.

Textwerte, NULL-Werte und fehlende Werte in einem oder beiden Teilen eines Wertepaars führen dazu, dass das Wertepaar ignoriert wird.

Siehe auch:

Beispiele zur Verwendung von linest-Funktionen (page 478)
Avg - Diagrammfunktion (page 419)

LINEST_SEY

LINEST_SEY() liefert den Standardfehler des geschätzten y-Wertes in der durch die Gleichung y=mx+b bestimmten linearen Regression für eine Reihe von Koordinaten, die durch x-expression und y-expression definiert sind, über die im **group by**-Zusatz bezeichneten Datensätze.

Syntax:

```
LINEST_SEY (y_value, x_value[, y0 [, x0 ]])
```

Rückgabe Datentyp: numerisch

Argumente:

Argument	Beschreibung
y_value	Die Formel oder das Feld mit dem Datenbereich y, der angegeben werden soll.
x_value	Die Formel oder das Feld mit dem Datenbereich x, der angegeben werden soll.
y(0), x(0)	Der optionale Parameter y0definiert, bei welchem Wert die Regressionsgerade die y-Achse schneidet. Die optionalen Parameter y0 und x0 definieren einen Koordinatenpunkt, durch den die Regressionsgerade geht. Sofern die Parameter y0 und x0 nicht definiert sind, erfordert die Berechnung zwei gültige Datenpaare. Wenn y0 und x0 definiert sind, reicht ein Datenpaar.

Beschränkungen:

Textwerte, NULL-Werte und fehlende Werte in einem oder beiden Teilen eines Wertepaars führen dazu, dass das Wertepaar ignoriert wird.

Siehe auch:

Beispiele zur Verwendung von linest-Funktionen (page 478)

LINEST_SEY - Diagrammfunktion

LINEST_SEY() liefert den Standardfehler des geschätzten y-Werts in der durch die Gleichung y=mx+b bestimmten linearen Regression für eine Reihe von Koordinaten, die durch **x_value** und **y_value** definiert sind, aggregiert nach den Dimensionen des Diagramms.

Syntax:

```
LINEST_SEY([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value[, y0_const[, x0_const]])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
y_value	Die Formel oder das Feld mit dem Datenbereich y, der angegeben werden soll.
x_value	Die Formel oder das Feld mit dem Datenbereich x, der angegeben werden soll.

Argument	Beschreibung
y0, x0	Der optionale Parameter y0definiert, bei welchem Wert die Regressionsgerade die y-Achse schneidet. Die optionalen Parameter y0 und x0 definieren einen Koordinatenpunkt, durch den die Regressionsgerade geht. *Sofern die Parameter y0 und x0 nicht definiert sind, erfordert die Berechnung zwei gültige Datenpaare. Wenn y0 und x0 definiert sind, reicht ein Datenpaar.*
SetExpression	Standardmäßig berechnet sich die Aggregierungsfunktion über alle wählbaren Werte. Alternativ können Sie die der Berechnung zugrunde liegenden Werte über die Auswahlformel bestimmen.
DISTINCT	Der Zusatz **DISTINCT** vor den Funktionsargumenten bewirkt, dass bei der Auswertung der Funktionsargumente entstehende Duplikate nicht berücksichtigt werden.
TOTAL	Der Zusatz **TOTAL** vor der Funktion bewirkt, dass die Berechnung über alle ausgewählten bzw. wählbaren Werte erfolgt, und nicht nur über diejenigen, die zu dem Wert der aktuellen Dimension zählen. Die Dimensionen des Diagramms werden also nicht berücksichtigt. Mit **TOTAL [<fld {.fld}>]**, wobei auf den Zusatz **TOTAL** eine Liste aus mindestens einem Feldnamen (d. h. einer Teilmenge der Diagrammdimensionsvariablen) folgt, erstellen Sie eine Teilmenge aller möglichen Werte.

Beschränkungen:

Die Parameter der Aggregierungsfunktion dürfen keine anderen Aggregierungsfunktionen enthalten, es sei denn, diese inneren Aggregierungen enthalten den Qualifizierer **TOTAL**. Für komplexere verschachtelte Aggregierungen verwenden Sie die erweiterte Funktion **Aggr** in Verbindung mit einer angegebenen Dimension.

Textwerte, NULL-Werte und fehlende Werte in einem oder beiden Teilen eines Wertepaars führen dazu, dass das Wertepaar ignoriert wird.

Siehe auch:

Beispiele zur Verwendung von linest-Funktionen (page 478)
Avg - Diagrammfunktion (page 419)

LINEST_SSREG

LINEST_SSREG() liefert die durch die Regression erklärte Varianz − Regression Sum of Squares − der durch die Gleichung y=mx+b bestimmten linearen Regression für eine Reihe von Koordinaten, die durch x-expression und y-expression definiert sind, aggregiert über eine Anzahl von Datensätzen, die durch eine **group by**-Bedingung definiert sind.

Syntax:

```
LINEST_SSREG (y_value, x_value[, y0 [, x0 ]])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
y_value	Die Formel oder das Feld mit dem Datenbereich y, der angegeben werden soll.
x_value	Die Formel oder das Feld mit dem Datenbereich x, der angegeben werden soll.
y(0), x(0)	Der optionale Parameter y0definiert, bei welchem Wert die Regressionsgerade die y-Achse schneidet. Die optionalen Parameter y0 und x0 definieren einen Koordinatenpunkt, durch den die Regressionsgerade geht. Sofern die Parameter y0 und x0 nicht definiert sind, erfordert die Berechnung zwei gültige Datenpaare. Wenn y0 und x0 definiert sind, reicht ein Datenpaar.

Beschränkungen:

Textwerte, NULL-Werte und fehlende Werte in einem oder beiden Teilen eines Wertepaars führen dazu, dass das Wertepaar ignoriert wird.

Siehe auch:

Beispiele zur Verwendung von linest-Funktionen (page 478)

LINEST_SSREG - Diagrammfunktion

LINEST_SSREG() liefert die durch die Regression erklärte Varianz, Regression Sum of Squares, in der durch die Gleichung y=mx+b bestimmten linearen Regression für eine Reihe von Koordinaten, die durch **x_value** und **y_value** definiert sind, aggregiert nach den Dimensionen des Diagramms.

Syntax:

```
LINEST_SSREG([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value[, y0_const[, x0_const]])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
y_value	Die Formel oder das Feld mit dem Datenbereich y, der angegeben werden soll.
x_value	Die Formel oder das Feld mit dem Datenbereich x, der angegeben werden soll.

Argument	Beschreibung
y0, x0	Der optionale Parameter y0definiert, bei welchem Wert die Regressionsgerade die y-Achse schneidet. Die optionalen Parameter y0 und x0 definieren einen Koordinatenpunkt, durch den die Regressionsgerade geht. *Sofern die Parameter y0 und x0 nicht definiert sind, erfordert die Berechnung zwei gültige Datenpaare. Wenn y0 und x0 definiert sind, reicht ein Datenpaar.*
SetExpression	Standardmäßig berechnet sich die Aggregierungsfunktion über alle wählbaren Werte. Alternativ können Sie die der Berechnung zugrunde liegenden Werte über die Auswahlformel bestimmen.
DISTINCT	Der Zusatz **DISTINCT** vor den Funktionsargumenten bewirkt, dass bei der Auswertung der Funktionsargumente entstehende Duplikate nicht berücksichtigt werden.
TOTAL	Der Zusatz **TOTAL** vor der Funktion bewirkt, dass die Berechnung über alle ausgewählten bzw. wählbaren Werte erfolgt, und nicht nur über diejenigen, die zu dem Wert der aktuellen Dimension zählen. Die Dimensionen des Diagramms werden also nicht berücksichtigt. Mit **TOTAL [<fld {.fld}>]**, wobei auf den Zusatz **TOTAL** eine Liste aus mindestens einem Feldnamen (d. h. einer Teilmenge der Diagrammdimensionsvariablen) folgt, erstellen Sie eine Teilmenge aller möglichen Werte.

Beschränkungen:

Die Parameter der Aggregierungsfunktion dürfen keine anderen Aggregierungsfunktionen enthalten, es sei denn, diese inneren Aggregierungen enthalten den Qualifizierer **TOTAL**. Für komplexere verschachtelte Aggregierungen verwenden Sie die erweiterte Funktion **Aggr** in Verbindung mit einer angegebenen Dimension.

Textwerte, NULL-Werte und fehlende Werte in einem oder beiden Teilen eines Wertepaars führen dazu, dass das Wertepaar ignoriert wird.

Siehe auch:

Beispiele zur Verwendung von linest-Funktionen (page 478)
Avg - Diagrammfunktion (page 419)

LINEST_SSRESID

LINEST_SSRESID() liefert die durch die residuale oder nicht erklärte Varianz, Residual Sum of Squares, der durch die Gleichung y=mx+b bestimmten linearen Regression für eine Reihe von Koordinaten, die durch x-expression und y-expression definiert sind, über die im **group by**-Zusatz bezeichneten Datensätze.

Syntax:

```
LINEST_SSRESID (y_value, x_value[, y0 [, x0 ]])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
y_value	Die Formel oder das Feld mit dem Datenbereich y, der angegeben werden soll.
x_value	Die Formel oder das Feld mit dem Datenbereich x, der angegeben werden soll.
y(0), x(0)	Der optionale Parameter y0definiert, bei welchem Wert die Regressionsgerade die y-Achse schneidet. Die optionalen Parameter y0 und x0 definieren einen Koordinatenpunkt, durch den die Regressionsgerade geht. Sofern die Parameter y0 und x0 nicht definiert sind, erfordert die Berechnung zwei gültige Datenpaare. Wenn y0 und x0 definiert sind, reicht ein Datenpaar.

Beschränkungen:

Textwerte, NULL-Werte und fehlende Werte in einem oder beiden Teilen eines Wertepaars führen dazu, dass das Wertepaar ignoriert wird.

Siehe auch:

Beispiele zur Verwendung von linest-Funktionen (page 478)

LINEST_SSRESID - Diagrammfunktion

LINEST_SSRESID() liefert die residuale oder nicht erklärte Varianz, Residual Sum of Squares, der durch die Gleichung y=mx+b bestimmten linearen Regression für eine Reihe von Koordinaten, die durch die Formeln **x_value** und **y_value** definiert sind, aggregiert nach den Dimensionen des Diagramms.

Syntax:

```
LINEST_SSRESID([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value,
x_value[, y0_const[, x0_const]])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
y_value	Die Formel oder das Feld mit dem Datenbereich y, der angegeben werden soll.
x_value	Die Formel oder das Feld mit dem Datenbereich x, der angegeben werden soll.

Argument	Beschreibung
y0, x0	Der optionale Parameter y0definiert, bei welchem Wert die Regressionsgerade die y-Achse schneidet. Die optionalen Parameter y0 und x0 definieren einen Koordinatenpunkt, durch den die Regressionsgerade geht. *Sofern die Parameter y0 und x0 nicht definiert sind, erfordert die Berechnung zwei gültige Datenpaare. Wenn y0 und x0 definiert sind, reicht ein Datenpaar.*
SetExpression	Standardmäßig berechnet sich die Aggregierungsfunktion über alle wählbaren Werte. Alternativ können Sie die der Berechnung zugrunde liegenden Werte über die Auswahlformel bestimmen.
DISTINCT	Der Zusatz **DISTINCT** vor den Funktionsargumenten bewirkt, dass bei der Auswertung der Funktionsargumente entstehende Duplikate nicht berücksichtigt werden.
TOTAL	Der Zusatz **TOTAL** vor der Funktion bewirkt, dass die Berechnung über alle ausgewählten bzw. wählbaren Werte erfolgt, und nicht nur über diejenigen, die zu dem Wert der aktuellen Dimension zählen. Die Dimensionen des Diagramms werden also nicht berücksichtigt. Mit **TOTAL [<fld {.fld}>]**, wobei auf den Zusatz **TOTAL** eine Liste aus mindestens einem Feldnamen (d. h. einer Teilmenge der Diagrammdimensionsvariablen) folgt, erstellen Sie eine Teilmenge aller möglichen Werte.

Der optionale Parameter y0definiert, bei welchem Wert die Regressionsgerade die y-Achse schneidet. Die optionalen Parameter y0 und x0 definieren einen Koordinatenpunkt, durch den die Regressionsgerade geht.

Beschränkungen:

Die Parameter der Aggregierungsfunktion dürfen keine anderen Aggregierungsfunktionen enthalten, es sei denn, diese inneren Aggregierungen enthalten den Qualifizierer **TOTAL**. Für komplexere verschachtelte Aggregierungen verwenden Sie die erweiterte Funktion **Aggr** in Verbindung mit einer angegebenen Dimension.

Textwerte, NULL-Werte und fehlende Werte in einem oder beiden Teilen eines Wertepaars führen dazu, dass das Wertepaar ignoriert wird.

Siehe auch:

- *Beispiele zur Verwendung von linest-Funktionen (page 478)*
- *Avg - Diagrammfunktion (page 419)*

Median

Median() liefert den aggregierten Median der Werte in der Formel über mehrere Datensätze, wie durch eine **group by**-Bedingung definiert wurde.

Syntax:

```
Median (expr)
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
expr	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.

Beispiel: Skriptformel mit Median

Beispiel – Skriptformel

Ladeskript

Laden Sie für dieses Beispiel die folgenden Inline-Daten und Skriptformeln in den Dateneditor.

```
Table 1:
Load RecNo() as RowNo, Letter, Number Inline
[Letter, Number
A,1
A,3
A,4
A,9
B,2
B,8
B,9];

Median:
LOAD Letter,
Median(Number) as MyMedian
Resident Table1 Group By Letter;
```

Erstellen einer Visualisierung

Erstellen Sie eine Tabellenvisualisierung in einem Qlik Sense Arbeitsblatt mit **Letter** und **MyMedian** als Dimensionen.

Ergebnis

Letter	MyMedian
A	3.5
B	8

Erläuterung

Der Median ist die „mittlere“ Zahl, wenn die Zahlen von der kleinsten zur größten sortiert wurden. Wenn der Datensatz eine gerade Anzahl Werte enthält, gibt die Funktion den Durchschnitt der beiden mittleren Werte zurück. In diesem Beispiel wird der Median für jeden Satz an Werten von **A** und **B** berechnet, also 3,5 bzw. 8.

Median - Diagrammfunktion

Median() liefert den Median des Bereichs von Werten, die in der Formel über die Dimensionen des Diagramms aggregiert wurden.

Syntax:

```
Median([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr)
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
expr	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.
SetExpression	Standardmäßig berechnet sich die Aggregierungsfunktion über alle wählbaren Werte. Alternativ können Sie die der Berechnung zugrunde liegenden Werte über die Auswahlformel bestimmen.
DISTINCT	Der Zusatz **DISTINCT** vor den Funktionsargumenten bewirkt, dass bei der Auswertung der Funktionsargumente entstehende Duplikate nicht berücksichtigt werden.
TOTAL	Der Zusatz **TOTAL** vor der Funktion bewirkt, dass die Berechnung über alle ausgewählten bzw. wählbaren Werte erfolgt, und nicht nur über diejenigen, die zu dem Wert der aktuellen Dimension zählen. Die Dimensionen des Diagramms werden also nicht berücksichtigt. Mit **TOTAL [<fld {.fld}>]**, wobei auf den Zusatz **TOTAL** eine Liste aus mindestens einem Feldnamen (d. h. einer Teilmenge der Diagrammdimensionsvariablen) folgt, erstellen Sie eine Teilmenge aller möglichen Werte.

Beschränkungen:

Die Parameter der Aggregierungsfunktion dürfen keine anderen Aggregierungsfunktionen enthalten, es sei denn, diese inneren Aggregierungen enthalten den Qualifizierer **TOTAL**. Für komplexere verschachtelte Aggregierungen verwenden Sie die erweiterte Funktion **Aggr** in Verbindung mit einer angegebenen Dimension.

Beispiel: Diagrammformel mit Median

Beispiel – Diagrammformel

Ladeskript

Laden Sie die folgenden Daten als Inline-Ladevorgang in den Dateneditor, um das folgende Diagrammformelbeispiel zu erstellen.

```
Load RecNo() as RowNo, Letter, Number Inline
[Letter, Number
A,1
A,3
```

```
A,4
A,9
B,2
B,8
B,9];
```

Erstellen einer Visualisierung

Erstellen Sie eine Tabellenvisualisierung in einem Qlik Sense Arbeitsblatt mit **Letter** als Dimension.

Diagrammformel

Fügen Sie die folgende Formel als Kennzahl zur Tabelle hinzu.

```
Median(Number)
```

Ergebnis

Letter	Median(Number)
Totals	4
A	3.5
B	8

Erläuterung

Der Median ist die „mittlere“ Zahl, wenn die Zahlen von der kleinsten zur größten sortiert wurden. Wenn der Datensatz eine gerade Anzahl Werte enthält, gibt die Funktion den Durchschnitt der beiden mittleren Werte zurück. In diesem Beispiel wird der Median für jeden Satz an Werten von **A** und **B** berechnet, also 3,5 bzw. 8.

Der Median für **Gesamtwerte** wird aus allen Werten berechnet und ergibt 4.

Siehe auch:

Avg - Diagrammfunktion (page 419)

MutualInfo - Diagrammfunktion

MutualInfo berechnet die gegenseitigen Informationen zwischen zwei Feldern oder zwischen aggregierten Werten in **Aggr()**.

MutualInfo liefert die aggregierten gegenseitigen Informationen für zwei Datensätze. Das ermöglicht eine Haupttreiberanalyse zwischen einem Feld und einem potenziellen Treiber. „Gegenseitige Informationen“ misst die Beziehung zwischen den Datensätzen und wird für Wertepaare (x,y) aggregiert, die Diagrammdimensionen durchlaufen. „Gegenseitige Informationen“ werden zwischen 0 und 1 gemessen und können als Perzentilwert formatiert werden. **MutualInfo** ist durch Auswahlen oder durch eine Auswahlformel definiert.

MutualInfo ermöglicht verschiedene Arten von MI-Analysen:

- Paarweise MI: Berechnen Sie die MI zwischen einem Treiberfeld und einem Zielfeld.
- Treiberaufschlüsselung nach Wert: Die MI wird zwischen einzelnen Feldwerten im Treiber- und im Zielfeld berechnet.
- Funktionsauswahl: Verwenden Sie **MutualInfo** in einem Matrixdiagramm, um eine Matrix zu generieren, in der alle Felder basierend auf MI miteinander verglichen werden.

MutualInfo gibt nicht unbedingt Kausalität zwischen Feldern an, die gegenseitige Informationen teilen. Zwei Felder können gegenseitige Informationen teilen, aber keine gleichwertigen Treiber füreinander sein. Wenn beispielsweise der Eisverkauf und die Außentemperaturen verglichen werden, zeigt **MutualInfo** die gegenseitigen Informationen zwischen den beiden. Dabei wird nicht angegeben, ob die Außentemperatur den Eisverkauf steigert, was wahrscheinlich ist, oder ob der Eisverkauf die Außentemperatur steigert, was unwahrscheinlich ist.

Wenn gegenseitige Informationen berechnet werden, wirken sich Verknüpfungen auf die Entsprechung zwischen und die Häufigkeit von Werten aus Feldern aus, die aus verschiedenen Tabellen stammen.

Zurückgegebene Werte für die gleichen Felder oder Auswahlen können leicht voneinander abweichen. Das liegt daran, dass jeder **MutualInfo**-Aufruf an einem zufällig ausgewählten Beispiel und mit der inhärenten Zufälligkeit des **MutualInfo**-Algorithmus durchgeführt wird.

MutualInfo kann auf die Funktion **Aggr()** angewendet werden.

Syntax:

```
MutualInfo({SetExpression}] [DISTINCT] [TOTAL] field1, field2 , datatype [,
breakdownbyvalue [, samplesize ]])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
field1, field2	Die Formeln oder Felder, die die beiden Beispielsätze enthalten, für die die gegenseitigen Informationen gemessen werden.
datatype	Die im Ziel und im Treiber enthaltenen Datentypen, 1 oder 'dd' für diskret:diskret 2 oder 'cc' für kontinuierlich:kontinuierlich 3 oder 'cd' für kontinuierlich:diskret 4 oder 'dc' für diskret:kontinuierlich Bei Datentypen wird nicht zwischen Groß- und Kleinschreibung unterschieden.

Argument	Beschreibung
breakdownbyvalue	Ein statischer Wert, der einem Wert im Treiber entspricht. Wenn er bereitgestellt wird, wird der MI-Beitrag für diesen Wert berechnet. Sie können **ValueList()** oder **ValueLoop()** verwenden. Wenn **Null()** hinzugefügt wird, wird die gesamte MI für alle Werte im Treiber berechnet. Für die Aufschlüsselung nach Wert ist es erforderlich, dass der Treiber diskrete Daten enthält.
samplesize	Die Anzahl der Werte, die vom Ziel und Treiber als Stichprobe genommen werden. Die Stichprobennahme erfolgt zufallsbasiert. **MutualInfo** erfordert eine Mindeststichprobengröße von 80. Standardmäßig nimmt **MutualInfo** nur eine Stichprobe von bis zu 10.000 Datenpaaren an, da **MutualInfo** ressourcenintensiv sein kann. Sie können eine größere Anzahl Datenpaare in der Stichprobengröße angeben. Reduzieren Sie im Fall einer Zeitüberschreitung von **MutualInfo** die Stichprobengröße.
SetExpression	Standardmäßig berechnet sich die Aggregierungsfunktion über alle wählbaren Werte. Alternativ können Sie die der Berechnung zugrunde liegenden Werte über die Auswahlformel bestimmen.
DISTINCT	Der Zusatz **DISTINCT** vor den Funktionsargumenten bewirkt, dass bei der Auswertung der Funktionsargumente entstehende Duplikate nicht berücksichtigt werden.
TOTAL	Der Zusatz **TOTAL** vor der Funktion bewirkt, dass die Berechnung über alle ausgewählten bzw. wählbaren Werte erfolgt, und nicht nur über diejenigen, die zu dem Wert der aktuellen Dimension zählen. Die Dimensionen des Diagramms werden also nicht berücksichtigt. Mit **TOTAL [<fld {.fld}>]**, wobei auf den Zusatz **TOTAL** eine Liste aus mindestens einem Feldnamen (d. h. einer Teilmenge der Diagrammdimensionsvariablen) folgt, erstellen Sie eine Teilmenge aller möglichen Werte.

Beschränkungen:

Textwerte, NULL-Werte und fehlende Werte in einem oder beiden Teilen eines Wertepaars führen dazu, dass das Wertepaar ignoriert wird.

Beispiele und Ergebnisse:

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Fügen Sie einem Arbeitsblatt in Ihrer App dann die Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um das Ergebnis anzuzeigen.

Funktionsbeispiele

Beispiel	Ergebnis
`mutualinfo (Age, Salary, 1)`	Für eine Tabelle mit der Dimension `Employee name` und der Kennzahl `mutualinfo(Age, Salary, 1)` ist das Ergebnis 0.99820986. Das Ergebnis wird nur für Zellen mit Gesamtwerten angezeigt.
`mutualinfo (TOTAL Age, Salary, 1, null(), 81)`	Wenn Sie ein Filterfenster mit der Dimension Gender erstellen und darin Auswahlen vornehmen, wird bei Auswahl von Female 0.99805677 und bei Auswahl von Male 0.99847373 ausgegeben. Dies geschieht, weil die Auswahl alle Ergebnisse ausschließt, die zum jeweils anderen Wert für Gender gehören.
`mutualinfo (TOTAL Age, Gender, 1, ValueLoop (25,35))`	0.68196996. Bei Auswahl eines beliebigen Werts in Gender ändert sich dies zu 0.
`mutualinfo ({1} TOTAL Age, Salary, 1, null())`	0.99820986. Dies ist unabhängig von Auswahlen. Die Auswahlformel {1} berücksichtigt keine Auswahlen und Dimensionen.

In Beispielen verwendete Daten:

```
Salary:

LOAD * inline [

"Employee name"|Age|Gender|Salary

Aiden Charles|20|Male|25000

Ann Lindquist|69|Female|58000

Anna Johansen|37|Female|36000

Anna Karlsson|42|Female|23000

Antonio Garcia|20|Male|61000

Benjamin Smith|42|Male|27000

Bill Yang|49|Male|50000

Binh Protzmann|69|Male|21000

Bob Park|51|Male|54000

Brenda Davies|25|Male|32000

Celine Gagnon|48|Female|38000
```

```
Cezar Sandu|50|Male|46000

Charles Ingvar Jönsson|27|Male|58000

Charlotte Edberg|45|Female|56000

Cindy Lynn|69|Female|28000

Clark Wayne|63|Male|31000

Daroush Ferrara|31|Male|29000

David Cooper|37|Male|64000

David Leg|58|Male|57000

Eunice Goldblum|31|Female|32000

Freddy Halvorsen|25|Male|26000

Gauri Indu|36|Female|46000

George van Zaant|59|Male|47000

Glenn Brown|58|Male|40000

Harry Jones|38|Male|40000

Helen Brolin|52|Female|66000

Hiroshi Ito|24|Male|42000

Ian Underwood|40|Male|45000

Ingrid Hendrix|63|Female|27000

Ira Baumel|39|Female|39000

Jackie Kingsley|23|Female|28000

Jennica Williams|36|Female|48000

Jerry Tessel|31|Male|57000

Jim Bond|50|Male|58000

Joan Callins|60|Female|65000

Joan Cleaves|25|Female|61000

Joe Cheng|61|Male|41000

John Doe|36|Male|59000
```

```
John Lemon|43|Male|21000

Karen Helmkey|54|Female|25000

Karl Berger|38|Male|68000

Karl Straubaum|30|Male|40000

Kaya Alpan|32|Female|60000

Kenneth Finley|21|Male|25000

Leif Shine|63|Male|70000

Lennart Skoglund|63|Male|24000

Leona Korhonen|46|Female|50000

Lina André|50|Female|65000

Louis Presley|29|Male|36000

Luke Langston|50|Male|63000

Marcus Salvatori|31|Male|46000

Marie Simon|57|Female|23000

Mario Rossi|39|Male|62000

Markus Danzig|26|Male|48000

Michael Carlen|21|Male|45000

Michelle Tyson|44|Female|69000

Mike Ashkenaz|45|Male|68000

Miro Ito|40|Male|39000

Nina Mihn|62|Female|57000

Olivia Nguyen|35|Female|51000

Olivier Simenon|44|Male|31000

Östen Ärlig|68|Male|57000

Pamala Garcia|69|Female|29000

Paolo Romano|34|Male|45000

Pat Taylor|67|Female|69000
```

```
Paul Dupont|34|Male|38000

Peter Smith|56|Male|53000

Pierre Clouseau|21|Male|37000

Preben Jørgensen|35|Male|38000

Rey Jones|65|Female|20000

Ricardo Gucci|55|Male|65000

Richard Ranieri|30|Male|64000

Rob Carsson|46|Male|54000

Rolf Wesenlund|25|Male|51000

Ronaldo Costa|64|Male|39000

Sabrina Richards|57|Female|40000

Sato Hiromu|35|Male|21000

Sehoon Daw|57|Male|24000

Stefan Lind|67|Male|35000

Steve Cioazzi|58|Male|23000

Sunil Gupta|45|Male|40000

Sven Svensson|45|Male|55000

Tom Lindwall|46|Male|24000

Tomas Nilsson|27|Male|22000

Trinity Rizzo|52|Female|48000

Vanessa Lambert|54|Female|27000

] (delimiter is '|');
```

Skew

Skew() liefert die Schiefe der Formel über die im **group by**-Zusatz bezeichneten Datensätze.

Syntax:

```
Skew([ distinct] expr)
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
expr	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.
DISTINCT	Ist der Formel das Wort **distinct** vorangestellt, werden Dubletten nicht berücksichtigt.

Beispiele und Ergebnisse:

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Bauen Sie dann ein Tabellendiagramm mit `Type` und *MySkew* als Dimensionen auf.

Ergebnisdaten

Beispiel	Ergebnis

`Table1:`
`Crosstable (Type, Value)`
`Load recno() as ID, * inline [`
`Observation|Comparison`
`35|2`
`40|27`
`12|38`
`15|31`
`21|1`
`14|19`
`46|1`
`10|34`
`28|3`
`48|1`
`16|2`
`30|3`
`32|2`
`48|1`
`31|2`
`22|1`
`12|3`
`39|29`
`19|37`
`25|2 ] (delimiter is '|');`
 	
`Skew1:`
`LOAD Type,`
`Skew(Value) as MySkew`
`Resident Table1 Group By Type;`
	Die Ergebnisse der Skew()-Berechnung:
	• `Type` ist `MySkew`
	• `Comparison` ist 0.86414768
	• `Observation` ist 0.32625351

Skew - Diagrammfunktion

Skew() liefert die Schiefe der Werte der Formel oder des Felds aggregiert nach den Dimensionen des Diagramms.

Syntax:

```
Skew([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr)
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
expr	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.
SetExpression	Standardmäßig berechnet sich die Aggregierungsfunktion über alle wählbaren Werte. Alternativ können Sie die der Berechnung zugrunde liegenden Werte über die Auswahlformel bestimmen.
DISTINCT	Der Zusatz **DISTINCT** vor den Funktionsargumenten bewirkt, dass bei der Auswertung der Funktionsargumente entstehende Duplikate nicht berücksichtigt werden.
TOTAL	Der Zusatz **TOTAL** vor der Funktion bewirkt, dass die Berechnung über alle ausgewählten bzw. wählbaren Werte erfolgt, und nicht nur über diejenigen, die zu dem Wert der aktuellen Dimension zählen. Die Dimensionen des Diagramms werden also nicht berücksichtigt. Mit **TOTAL [<fld {.fld}>]**, wobei auf den Zusatz **TOTAL** eine Liste aus mindestens einem Feldnamen (d. h. einer Teilmenge der Diagrammdimensionsvariablen) folgt, erstellen Sie eine Teilmenge aller möglichen Werte.

Beschränkungen:

Die Parameter der Aggregierungsfunktion dürfen keine anderen Aggregierungsfunktionen enthalten, es sei denn, diese inneren Aggregierungen enthalten den Qualifizierer **TOTAL**. Für komplexere verschachtelte Aggregierungen verwenden Sie die erweiterte Funktion **Aggr** in Verbindung mit einer angegebenen Dimension.

Beispiele und Ergebnisse:

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Bauen Sie dann ein Tabellendiagramm mit `Type` als Dimension und `Skew(value)` als Kennzahl auf.

`Totals` muss in den Tabelleneigenschaften aktiviert sein.

Beispiel	Ergebnis
Table1: Crosstable (Type, Value) Load recno() as ID, * inline [Observation|Comparison 35|2 40|27 12|38 15|31 21|1 14|19 46|1 10|34 28|3 48|1 16|2 30|3 32|2 48|1 31|2 22|1 12|3 39|29 19|37 25|2] (delimiter is '|');	Die Ergebnisse der Skew(Value)-Berechnung: • Total ist 0.23522195 • Comparison ist 0.86414768 • Observation ist 0.32625351

Siehe auch:

Avg - Diagrammfunktion (page 419)

Stdev

Stdev() liefert die Standardabweichung der in der Formel über mehrere Datensätze angegebenen Werte, wie in einer Bedingung **group by** definiert wurde.

Syntax:

```
Stdev([distinct] expr)
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
expr	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.
distinct	Ist der Formel das Wort **distinct** vorangestellt, werden Dubletten nicht berücksichtigt.

Beispiele und Ergebnisse:

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Bauen Sie dann ein Tabellendiagramm mit `Type` und `MyStdev` als Dimensionen auf.

Ergebnisdaten

Beispiel	Ergebnis

`Table1:`
`Crosstable (Type, Value)`
`Load recno() as ID, * inline [`
`Observation|Comparison`
`35|2`
`40|27`
`12|38`
`15|31`
`21|1`
`14|19`
`46|1`
`10|34`
`28|3`
`48|1`
`16|2`
`30|3`
`32|2`
`48|1`
`31|2`
`22|1`
`12|3`
`39|29`
`19|37`
`25|2 ] (delimiter is '|');`
 	
`Stdev1:`
`LOAD Type,`
`Stdev(Value) as MyStdev`
`Resident Table1 Group By Type;`
	Die Ergebnisse der Stdev()-Berechnung:
	• `Type` ist `MyStdev`
	• `Comparison` ist 14.61245
	• `Observation` ist 12.507997

Stdev - Diagrammfunktion

Stdev() sucht nach der Standardabweichung des in der Formel oder im Feld aggregierten Bereichs von Daten, die über die Dimensionen des Diagramms iteriert wurden.

Syntax:

```
Stdev([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr)
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
expr	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.
SetExpression	Standardmäßig berechnet sich die Aggregierungsfunktion über alle wählbaren Werte. Alternativ können Sie die der Berechnung zugrunde liegenden Werte über die Auswahlformel bestimmen.
DISTINCT	Der Zusatz **DISTINCT** vor den Funktionsargumenten bewirkt, dass bei der Auswertung der Funktionsargumente entstehende Duplikate nicht berücksichtigt werden.
TOTAL	Der Zusatz **TOTAL** vor der Funktion bewirkt, dass die Berechnung über alle ausgewählten bzw. wählbaren Werte erfolgt, und nicht nur über diejenigen, die zu dem Wert der aktuellen Dimension zählen. Die Dimensionen des Diagramms werden also nicht berücksichtigt. Mit **TOTAL [<fld {.fld}>]**, wobei auf den Zusatz **TOTAL** eine Liste aus mindestens einem Feldnamen (d. h. einer Teilmenge der Diagrammdimensionsvariablen) folgt, erstellen Sie eine Teilmenge aller möglichen Werte.

Beschränkungen:

Die Parameter der Aggregierungsfunktion dürfen keine anderen Aggregierungsfunktionen enthalten, es sei denn, diese inneren Aggregierungen enthalten den Qualifizierer **TOTAL**. Für komplexere verschachtelte Aggregierungen verwenden Sie die erweiterte Funktion **Aggr** in Verbindung mit einer angegebenen Dimension.

Beispiele und Ergebnisse:

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Bauen Sie dann ein Tabellendiagramm mit `Type` als Dimension und `Stdev(Value)` als Kennzahl auf.

`Totals` muss in den Tabelleneigenschaften aktiviert sein.

Beispiel	Ergebnis
Stdev(Value) Table1: Crosstable (Type, Value) Load recno() as ID, * inline [Observation|Comparison 35|2 40|27 12|38 15|31 21|1 14|19 46|1 10|34 28|3 48|1 16|2 30|3 32|2 48|1 31|2 22|1 12|3 39|29 19|37 25|2] (delimiter is '|');	Die Ergebnisse der Stdev(Value)-Berechnung: • Totalist 15.47529 • Comparison ist 14.61245 • Observation ist 12.507997

Siehe auch:

Avg - Diagrammfunktion (page 419)
STEYX - Diagrammfunktion (page 476)

Sterr

Sterr() liefert den Standardfehler (stdev/sqrt(n)) für eine Wertemenge einer Formel über die in der **group by**-Bedingung bezeichneten Datensätze.

Syntax:

```
Sterr ([distinct] expr)
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
expr	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.
distinct	Ist der Formel das Wort **distinct** vorangestellt, werden Dubletten nicht berücksichtigt.

Beschränkungen:

Textwerte, NULL-Werte und fehlende Werte werden ignoriert.

Beispiele und Ergebnisse:

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Fügen Sie einem Arbeitsblatt in Ihrer App dann die Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um das Ergebnis anzuzeigen.

Ergebnisdaten

Beispiel	Ergebnis

`Table1:`
`Crosstable (Type, Value)`
`Load recno() as ID, * inline [`
`Observation|Comparison`
`35|2`
`40|27`
`12|38`
`15|31`
`21|1`
`14|19`
`46|1`
`10|34`
`28|3`
`48|1`
`16|2`
`30|3`
`32|2`
`48|1`
`31|2`
`22|1`
`12|3`
`39|29`
`19|37`
`25|2 ] (delimiter is '|');`
 	
`Sterr1:`
`LOAD Type,`
`Sterr(Value) as MySterr`
`Resident Table1 Group By Type;`
	In einer Tabelle mit den Dimensionen `Type` und `MySterr` werden die Ergebnisse der Berechnung von Sterr() im Datenladeskript angezeigt:
	`Type MySterr`
	
	`Comparison 3.2674431`
	
	`Observation 2.7968733`

Sterr - Diagrammfunktion

Sterr() liefert den Wert des Standardfehlers des Mittels (stdev/sqrt(n)) für die Werte, die in der Formel über die Dimensionen des Diagramms aggregiert wurden.

Syntax:

```
Sterr([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr)
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
expr	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.
SetExpression	Standardmäßig berechnet sich die Aggregierungsfunktion über alle wählbaren Werte. Alternativ können Sie die der Berechnung zugrunde liegenden Werte über die Auswahlformel bestimmen.
DISTINCT	Der Zusatz **DISTINCT** vor den Funktionsargumenten bewirkt, dass bei der Auswertung der Funktionsargumente entstehende Duplikate nicht berücksichtigt werden.
TOTAL	Der Zusatz **TOTAL** vor der Funktion bewirkt, dass die Berechnung über alle ausgewählten bzw. wählbaren Werte erfolgt, und nicht nur über diejenigen, die zu dem Wert der aktuellen Dimension zählen. Die Dimensionen des Diagramms werden also nicht berücksichtigt. Mit **TOTAL [<fld {.fld}>]**, wobei auf den Zusatz **TOTAL** eine Liste aus mindestens einem Feldnamen (d. h. einer Teilmenge der Diagrammdimensionsvariablen) folgt, erstellen Sie eine Teilmenge aller möglichen Werte.

Beschränkungen:

Die Parameter der Aggregierungsfunktion dürfen keine anderen Aggregierungsfunktionen enthalten, es sei denn, diese inneren Aggregierungen enthalten den Qualifizierer **TOTAL**. Für komplexere verschachtelte Aggregierungen verwenden Sie die erweiterte Funktion **Aggr** in Verbindung mit einer angegebenen Dimension.

Textwerte, NULL-Werte und fehlende Werte werden ignoriert.

Beispiele und Ergebnisse:

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Bauen Sie dann ein Tabellendiagramm mit `Type` als Dimension und `Sterr(Value)` als Kennzahl auf.

`Totals` muss in den Tabelleneigenschaften aktiviert sein.

Beispiel	Ergebnis
Table1: Crosstable (Type, Value) Load recno() as ID, * inline [Observation|Comparison 35|2 40|27 12|38 15|31 21|1 14|19 46|1 10|34 28|3 48|1 16|2 30|3 32|2 48|1 31|2 22|1 12|3 39|29 19|37 25|2] (delimiter is '|');	Die Ergebnisse der Sterr(Value)-Berechnung: • Totalist 2.4468583 • Comparison ist 3.2674431 • Observation ist 2.7968733

Siehe auch:

- *Avg - Diagrammfunktion (page 419)*
- *STEYX - Diagrammfunktion (page 476)*

STEYX

STEYX() liefert den Standardfehler des geschätzten y-Werts für jeden x-Wert in der Regression für eine Reihe von Wertepaaren, die durch x-expression und y-expression definiert sind, über die im **group by**-Zusatz bezeichneten Datensätze.

Syntax:

```
STEYX (y_value, x_value)
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
y_value	Die Formel oder das Feld mit dem Datenbereich y, der angegeben werden soll.
x_value	Die Formel oder das Feld mit dem Datenbereich x, der angegeben werden soll.

Beschränkungen:

Textwerte, NULL-Werte und fehlende Werte in einem oder beiden Teilen eines Wertepaars führen dazu, dass das Wertepaar ignoriert wird.

Beispiele und Ergebnisse:

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Fügen Sie einem Arbeitsblatt in Ihrer App dann die Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um das Ergebnis anzuzeigen.

Ergebnisdaten

Beispiel	Ergebnis
Trend: Load *, 1 as Grp; LOAD * inline [Month|KnownY|KnownX Jan|2|6 Feb|3|5 Mar|9|11 Apr|6|7 May|8|5 Jun|7|4 Jul|5|5 Aug|10|8 Sep|9|10 Oct|12|14 Nov|15|17 Dec|14|16] (delimiter is '|'); STEYX1: LOAD Grp, STEYX(KnownY, KnownX) as MySTEYX Resident Trend Group By Grp;	In einer Tabelle mit der Dimension MySTEYX wird das Ergebnis der Berechnung von STEYX() im Datenladeskript angezeigt: 2.0714764.

STEYX - Diagrammfunktion

STEYX() liefert den Standardfehler der geschätzten y-Werte für jeden x-Wert in einer linearen Regression für eine Reihe von Koordinaten, die durch **y_value** und **x_value** definiert sind.

Syntax:

```
STEYX([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value)
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
y_value	Die Formel oder das Feld mit dem bekannten Datenbereich y, der angegeben werden soll.
x_value	Die Formel oder das Feld mit dem bekannten Datenbereich x, der angegeben werden soll.
SetExpression	Standardmäßig berechnet sich die Aggregierungsfunktion über alle wählbaren Werte. Alternativ können Sie die der Berechnung zugrunde liegenden Werte über die Auswahlformel bestimmen.
DISTINCT	Der Zusatz **DISTINCT** vor den Funktionsargumenten bewirkt, dass bei der Auswertung der Funktionsargumente entstehende Duplikate nicht berücksichtigt werden.
TOTAL	Der Zusatz **TOTAL** vor der Funktion bewirkt, dass die Berechnung über alle ausgewählten bzw. wählbaren Werte erfolgt, und nicht nur über diejenigen, die zu dem Wert der aktuellen Dimension zählen. Die Dimensionen des Diagramms werden also nicht berücksichtigt. Mit **TOTAL [<fld {.fld}>]**, wobei auf den Zusatz **TOTAL** eine Liste aus mindestens einem Feldnamen (d. h. einer Teilmenge der Diagrammdimensionsvariablen) folgt, erstellen Sie eine Teilmenge aller möglichen Werte.

Beschränkungen:

Die Parameter der Aggregierungsfunktion dürfen keine anderen Aggregierungsfunktionen enthalten, es sei denn, diese inneren Aggregierungen enthalten den Qualifizierer **TOTAL**. Für komplexere verschachtelte Aggregierungen verwenden Sie die erweiterte Funktion **Aggr** in Verbindung mit einer angegebenen Dimension.

Textwerte, NULL-Werte und fehlende Werte in einem oder beiden Teilen eines Wertepaars führen dazu, dass das Wertepaar ignoriert wird.

Beispiele und Ergebnisse:

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Bauen Sie dann ein Tabellendiagramm mit `KnownY` und `KnownX` als Dimension und `Steyx(KnownY,KnownX)` als Kennzahl.

`Totals` muss in den Tabelleneigenschaften aktiviert sein.

Beispiel	Ergebnis
Trend: LOAD * inline [Month|KnownY|KnownX Jan|2|6 Feb|3|5 Mar|9|11 Apr|6|7 May|8|5 Jun|7|4 Jul|5|5 Aug|10|8 Sep|9|10 Oct|12|14 Nov|15|17 Dec|14|16] (delimiter is '|');	Das Ergebnis der Berechnung STEYX(KnownY,KnownX) ist 2.071 (wenn die Zahlenformatierung auf 3 Dezimalstellen beschränkt ist)

Siehe auch:

- *Avg - Diagrammfunktion (page 419)*
- *Sterr - Diagrammfunktion (page 472)*

Beispiele zur Verwendung von linest-Funktionen

Die Funktionen linest werden zum Ermitteln von Werten verwendet, die mit der Analyse der linearen Regression verbunden sind. In diesem Abschnitt wird das Erstellen von Visualisierungen mithilfe von Beispieldaten beschrieben. Dies ermöglicht das Ermitteln der Werte von linest-Funktionen in Qlik Sense. Die linest-Funktionen können sowohl im Datenladeskript als auch in den Diagrammformeln verwendet werden.

Beschreibungen zu Syntax und Argumenten sind in den einzelnen Themen zu den Diagramm- und Skriptfunktionen für linest zu finden.

In den Beispielen verwendete Daten und Skriptformeln

Laden Sie die folgenden Inline-Daten und Skriptformeln in den Dateneditor, um die folgenden Beispiele für linest() zu erstellen.

```
T1:
LOAD *, 1 as Grp;
LOAD * inline [
X|Y
1|0
2|1
3|3
4|8
5|14
6|20
7|0
8|50
9|25
10|60
11|38
12|19
13|26
14|143
15|98
16|27
17|59
18|78
19|158
20|279 ] (delimiter is '|');

R1:
LOAD
Grp,
linest_B(Y,X) as Linest_B,
linest_DF(Y,X) as Linest_DF,
linest_F(Y,X) as Linest_F,
linest_M(Y,X) as Linest_M,
linest_R2(Y,X) as Linest_R2,
linest_SEB(Y,X,1,1) as Linest_SEB,
linest_SEM(Y,X) as Linest_SEM,
linest_SEY(Y,X) as Linest_SEY,
linest_SSREG(Y,X) as Linest_SSREG,
linest_SSRESID(Y,X) as Linest_SSRESID
resident T1 group by Grp;
```

Beispiel 1: Skriptformeln mit linest

Beispiel: Formeln im Skript

Erstellen Sie eine Visualisierung anhand der Berechnungen des Datenladeskripts

Erstellen Sie eine Tabellenvisualisierung in einem Qlik Sense Arbeitsblatt mit den folgenden Feldern als Spalten:

- Linest_B
- Linest_DF

- Linest_F
- Linest_M
- Linest_R2
- Linest_SEB
- Linest_SEM
- Linest_SEY
- Linest_SSREG
- Linest_SSRESID

Ergebnis
Die Tabelle mit den Ergebnissen der im Datenladeskript angestellten linest-Berechnungen sollte folgendermaßen aussehen:

Ergebnistabelle

Linest_B	Linest_DF	Linest_F	Linest_M	Linest_R2	Linest_SEB
-35.047	18	20.788	8.605	0.536	22.607

Ergebnistabelle

Linest_SEM	Linest_SEY	Linest_SSREG	Linest_SSRESID
1.887	48.666	49235.014	42631.186

Beispiel 2: Diagrammformeln mit linest

Beispiel: Diagrammformeln
Erstellen Sie eine Tabellenvisualisierung in einem Qlik Sense Arbeitsblatt mit den folgenden Feldern als Dimensionen:

```
ValueList('Linest_b', 'Linest_df','Linest_f', 'Linest_m','Linest_r2','Linest_SEB','Linest_
SEM','Linest_SEY','Linest_SSREG','Linest_SSRESID')
```

Die Formel verwendet die Funktion für synthetische Dimensionen, um Beschriftungen für die Dimensionen mit den Namen der linest-Funktionen zu erstellen. Sie können die Beschriftung der **Linest functions** ändern, um Platz zu sparen.

Fügen Sie die folgende Formel als Kennzahl zur Tabelle hinzu:

```
Pick(Match(ValueList('Linest_b', 'Linest_df','Linest_f', 'Linest_m','Linest_r2','Linest_
SEB','Linest_SEM','Linest_SEY','Linest_SSREG','Linest_SSRESID'),'Linest_b', 'Linest_
df','Linest_f', 'Linest_m','Linest_r2','Linest_SEB','Linest_SEM','Linest_SEY','Linest_
```

```
SSREG','Linest_SSRESID'),Linest_b(Y,X),Linest_df(Y,X),Linest_f(Y,X),Linest_m(Y,X),Linest_r2
(Y,X),Linest_SEB(Y,X,1,1),Linest_SEM(Y,X),Linest_SEY(Y,X),Linest_SSREG(Y,X),Linest_SSRESID
(Y,X) )
```

Diese Formel zeigt den Wert des Ergebnisses jeder linest-Funktion mit dem entsprechenden Namen in der synthetischen Dimension an. Das Ergebnis von `Linest_b(Y,X)` wird neben **linest_b** angezeigt, usw.

Ergebnis

Ergebnistabelle

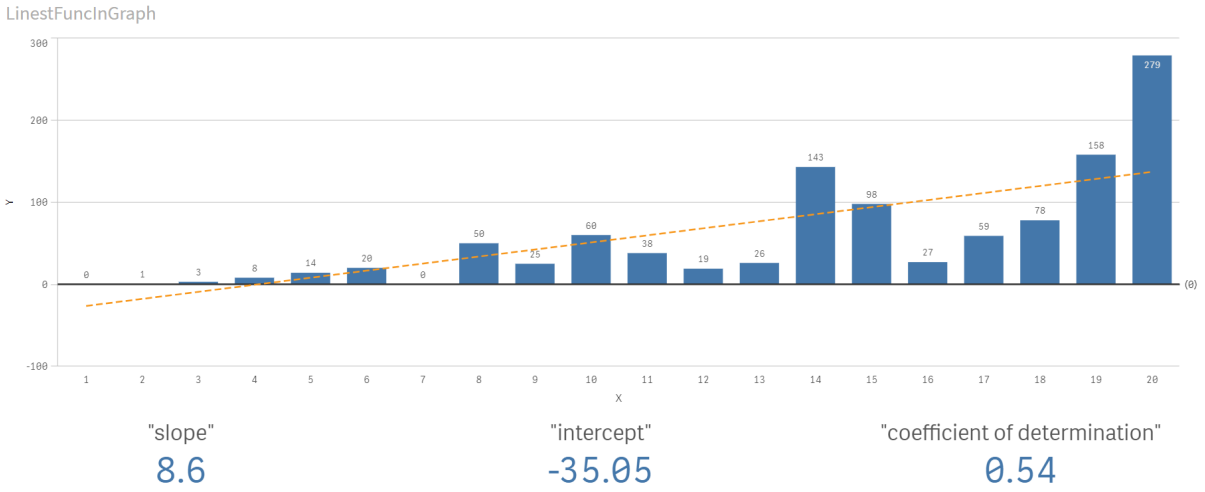
Linest functions	**Linest function results**
Linest_b	-35.047
Linest_df	18
Linest_f	20.788
Linest_m	8.605
Linest_r2	0.536
Linest_SEB	22.607
Linest_SEM	1.887
Linest_SEY	48.666
Linest_SSREG	49235.014
Linest_SSRESID	42631.186

Beispiel 3: Diagrammformeln mit linest

Beispiel: Diagrammformeln

1. Erstellen Sie eine Balkendiagramm-Visualisierung in einem Qlik Sense Arbeitsblatt mit **X** als Dimension und **Y** als Kennzahl.
2. Fügen Sie eine lineare Trendlinie zur Kennzahl Y hinzu.
3. Fügen Sie dem Arbeitsblatt eine KPI-Visualisierung hinzu.
    1. Fügen Sie *slope* als Beschriftung für den KPI hinzu.
    2. Fügen Sie `sum(Linest_M)` als Formel für den KPI hinzu.
4. Fügen Sie dem Arbeitsblatt eine zweite KPI-Visualisierung hinzu.
    1. Fügen Sie *intercept* als Beschriftung für den KPI hinzu.
    2. Fügen Sie `Sum(Linest_B)` als Formel für den KPI hinzu.
5. Fügen Sie dem Arbeitsblatt eine dritte KPI-Visualisierung hinzu.
    1. Fügen Sie *coefficient of determination* als Beschriftung für den KPI hinzu.
    2. Fügen Sie `Sum(Linest_R2)` als Formel für den KPI hinzu.

Ergebnis



"slope"
8.6

"intercept"
-35.05

"coefficient of determination"
0.54

Erläuterung

Das Balkendiagramm zeigt die Darstellung der X- und Y-Daten. Relevante linest()-Funktionen stellen Werte für die lineare Regressionsgleichung bereit, auf der die Trendlinie basiert, d. h. y = m * x + b. Die Gleichung verwendet die Methode der kleinsten Quadrate zum Berechnen einer geraden Linie (Trendlinie) und gibt ein Array zurück, das die am besten zu den Daten passende Linie beschreibt.

Die KPIs zeigen die Ergebnisse der linest()-Funktionen **sum(Linest_M)** für den Anstieg und **sum(Linest_B)** für den Y-Achsenabschnitt (die Variablen in der linearen Regressionsgleichung) und den entsprechenden aggregierten R2-Wert für den Determinationskoeffizienten an.

Funktionen für statistische Tests

Statistische Testfunktionen können sowohl im Datenladeskript als auch in den Diagrammformeln verwendet werden, die Syntax ist jedoch unterschiedlich.

Chi-2-Testfunktionen

Wird in der Regel in der Studie von qualitativen Variablen verwendet. Es kann zum einen die Verteilung in einer einfachen Häufigkeitstabelle mit erwarteter Häufigkeit verglichen werden oder es kann die Verbindung zwischen zwei Variablen in einer Kontingenztabelle untersucht werden.

T-Testfunktionen

T-Testfunktionen werden zur statistischen Untersuchung zweier Populationen verwendet. Ein t-Test von zwei Stichproben untersucht, ob die beiden Proben sich unterscheiden. Er wird in der Regel verwendet, wenn zwei normale Verteilungen über unbekannte Varianzen verfügen und es sich um ein Experiment mit kleinem Stichprobenumfang handelt.

Z-Testfunktionen

Eine statistische Untersuchung zweier Populationsmittel. Ein z-Test von zwei Stichproben untersucht, ob die beiden Proben sich unterscheiden. Er wird in der Regel verwendet, wenn zwei normale Verteilungen über bekannte Varianzen verfügen und es sich um ein Experiment mit großem Stichprobenumfang handelt.

Chi2-Testfunktionen

Wird in der Regel in der Studie von qualitativen Variablen verwendet. Es kann zum einen die Verteilung in einer einfachen Häufigkeitstabelle mit erwarteter Häufigkeit verglichen werden oder es kann die Verbindung zwischen zwei Variablen in einer Kontingenztabelle untersucht werden.Chi-squared test functions are used to determine whether there is a statistically significant difference between the expected frequencies and the observed frequencies in one or more groups. Often a histogram is used, and the different bins are compared to an expected distribution.

Falls die Funktion im Datenladeskript verwendet wird, werden die Werte über mehrere im group by-Zusatz bezeichnete Datensätze aggregiert.

Wird die Funktion als Diagrammformel verwendet, werden die Werte über die Diagrammdimensionen aggregiert.

Chi2Test_chi2
Chi2Test_chi2() liefert den aggregierten Wert für den Chi^2-Test einer oder zweier Wertemengen.

```
Chi2Test_chi2() liefert den aggregierten Wert für den Chi2-Test einer oder zweier Wertemengen.(col, row, actual_value[, expected_value])
```

Chi2Test_df
Chi2Test_df() liefert den Wert df (Freiheitsgrade) für den Chi^2-Test einer oder zweier Wertemengen.

```
Chi2Test_df() liefert den Wert df (Freiheitsgrade) für den Chi2-Test einer oder zweier Wertemengen.(col, row, actual_value[, expected_value])
```

Chi2Test_p
Chi2Test_p() liefert den Wert p (Signifikanz) für den Chi^2-Test einer oder zweier Wertemengen.

```
Chi2Test_p - Diagrammfunktion(col, row, actual_value[, expected_value])
```

Siehe auch:

T-Testfunktionen (page 486)
Z-Testfunktionen (page 521)

Chi2Test_chi2

Chi2Test_chi2() liefert den aggregierten Wert für den Chi^2-Test einer oder zweier Wertemengen.

Falls die Funktion im Datenladeskript verwendet wird, werden die Werte über mehrere im group by-Zusatz bezeichnete Datensätze aggregiert.

Wird die Funktion als Diagrammformel verwendet, werden die Werte über die Diagrammdimensionen aggregiert.

Alle Qlik Sense chi^2-Testfunktionen verfügen über dieselben Argumente.

Syntax:

```
Chi2Test_chi2(col, row, actual_value[, expected_value])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
col, row	Die angegebene Spalte und Zeile in der Matrix der zu testenden Werte.
actual_value	Der Schätzwert der in **col** und **row** angegebenen Daten.
expected_value	Der erwartete Wert für die in **col** und **row** angegebene Verteilung.

Beschränkungen:

Bei Textwerten, NULL-Werten oder fehlenden Werten im Formelwert liefert die Funktion NULL.

Beispiele:

```
Chi2Test_chi2( Grp, Grade, Count )
Chi2Test_chi2( Gender, Description, Observed, Expected )
```

Siehe auch:

Beispiele zur Verwendung der chi2-test-Funktionen in Diagrammen (page 537)
Beispiele zur Verwendung der chi2-test-Funktionen im Datenladeskript (page 541)

Chi2Test_df

Chi2Test_df() liefert den Wert df (Freiheitsgrade) für den Chi^2-Test einer oder zweier Wertemengen.

Falls die Funktion im Datenladeskript verwendet wird, werden die Werte über mehrere im group by-Zusatz bezeichnete Datensätze aggregiert.

Wird die Funktion als Diagrammformel verwendet, werden die Werte über die Diagrammdimensionen aggregiert.

Alle Qlik Sense chi^2-Testfunktionen verfügen über dieselben Argumente.

Syntax:

```
Chi2Test_df(col, row, actual_value[, expected_value])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
col, row	Die angegebene Spalte und Zeile in der Matrix der zu testenden Werte.
actual_value	Der Schätzwert der in **col** und **row** angegebenen Daten.
expected_value	Der erwartete Wert für die in **col** und **row** angegebene Verteilung.

Beschränkungen:

Bei Textwerten, NULL-Werten oder fehlenden Werten im Formelwert liefert die Funktion NULL.

Beispiele:

```
Chi2Test_df( Grp, Grade, Count )
Chi2Test_df( Gender, Description, Observed, Expected )
```

Siehe auch:

Beispiele zur Verwendung der chi2-test-Funktionen in Diagrammen (page 537)
Beispiele zur Verwendung der chi2-test-Funktionen im Datenladeskript (page 541)

Chi2Test_p - Diagrammfunktion

Chi2Test_p() liefert den Wert p (Signifikanz) für den Chi^2-Test einer oder zweier Wertemengen. Der Test bezieht sich entweder auf die Werte in **actual_value** und prüft auf Abweichungen innerhalb der durch **col** und **row** angegebenen Wertematrix, oder er vergleicht die Werte in **actual_value** mit den zugehörigen Werten in **expected_value**.

Falls die Funktion im Datenladeskript verwendet wird, werden die Werte über mehrere im group by-Zusatz bezeichnete Datensätze aggregiert.

Wird die Funktion als Diagrammformel verwendet, werden die Werte über die Diagrammdimensionen aggregiert.

Alle Qlik Sense chi^2-Testfunktionen verfügen über dieselben Argumente.

Syntax:

```
Chi2Test_p(col, row, actual_value[, expected_value])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
col, row	Die angegebene Spalte und Zeile in der Matrix der zu testenden Werte.
actual_value	Der Schätzwert der in **col** und **row** angegebenen Daten.
expected_value	Der erwartete Wert für die in **col** und **row** angegebene Verteilung.

Beschränkungen:

Bei Textwerten, NULL-Werten oder fehlenden Werten im Formelwert liefert die Funktion NULL.

Beispiele:

```
Chi2Test_p( Grp, Grade, Count )
Chi2Test_p( Gender, Description, Observed, Expected )
```

Siehe auch:

Beispiele zur Verwendung der chi2-test-Funktionen in Diagrammen (page 537)
Beispiele zur Verwendung der chi2-test-Funktionen im Datenladeskript (page 541)

T-Testfunktionen

T-Testfunktionen werden zur statistischen Untersuchung zweier Populationen verwendet. Ein t-Test von zwei Stichproben untersucht, ob die beiden Proben sich unterscheiden. Er wird in der Regel verwendet, wenn zwei normale Verteilungen über unbekannte Varianzen verfügen und es sich um ein Experiment mit kleinem Stichprobenumfang handelt.

In den folgenden Abschnitten sind die statistischen t-Testfunktionen nach dem Schülerstichprobentest gruppiert, der sich auf jede Funktionsart bezieht.

Erstellen eines typischen t-test-Reports (page 543)

T-Tests mit zwei unabhängigen Stichproben

Die folgenden Funktionen beziehen sich auf zwei unabhängige Stichproben:

ttest_conf

TTest_conf liefert den aggregierten t-Test-Wert des Konfidenzintervalls zweier unabhängiger Stichproben.

```
TTest_conf liefert den aggregierten t-Test-Wert des Konfidenzintervalls
zweier unabhängiger Stichproben. ( grp, value [, sig[, eq_var]])
```

ttest_df

TTest_df() liefert den aggregierten Wert (Freiheitsgrade) für den Studenten-t-Test zweier unabhängiger Stichproben.

```
TTest_df() liefert den aggregierten Wert (Freiheitsgrade) für den Studenten-t-Test zweier unabhängiger Stichproben. (grp, value [, eq_var)
```

ttest_dif

TTest_dif() ist eine numerische Funktion, die die aggregierte Mittelwertdifferenz für den Studenten-t-Test zweier unabhängiger Stichproben liefert.

```
TTest_dif() ist eine numerische Funktion, die die aggregierte Mittelwertdifferenz für den Studenten-t-Test zweier unabhängiger Stichproben liefert. (grp, value)
```

ttest_lower

TTest_lower() liefert den aggregierten Wert für das untere Ende des Konfidenzintervalls zweier unabhängiger Stichproben.

```
TTest_lower() liefert den aggregierten Wert für das untere Ende des Konfidenzintervalls zweier unabhängiger Stichproben. (grp, value [, sig[, eq_var]])
```

ttest_sig

TTest_sig() liefert das 2-seitige Signifikanzniveau für den aggregierten Studenten-t-Test zweier unabhängiger Stichproben.

```
TTest_sig() liefert das 2-seitige Signifikanzniveau für den aggregierten Studenten-t-Test zweier unabhängiger Stichproben. (grp, value [, eq_var])
```

ttest_sterr

TTest_sterr() liefert den Standardfehler der Mittelwertdifferenz für den aggregierten Studenten-t-Test zweier unabhängiger Stichproben.

```
TTest_sterr() liefert den Standardfehler der Mittelwertdifferenz für den aggregierten Studenten-t-Test zweier unabhängiger Stichproben. (grp, value [, eq_var])
```

ttest_t

TTest_t() liefert den aggregierten t-Wert zweier unabhängiger Stichproben.

```
TTest_t() liefert den aggregierten t-Wert zweier unabhängiger Stichproben. (grp, value [, eq_var])
```

ttest_upper

TTest_upper() liefert den aggregierten Wert für das obere Ende des Konfidenzintervalls zweier unabhängiger Stichproben.

TTest_upper() liefert den aggregierten Wert für das obere Ende des Konfidenzintervalls zweier unabhängiger Stichproben. (grp, value [, sig [, eq_var]]**)**

T-Tests mit zwei unabhängigen, gewichteten Stichproben

Die folgenden Funktionen beziehen sich auf den t-Test zweier unabhängiger Stichproben, bei denen die Werte der Stichproben unterschiedlich gewichtet sind.

ttestw_conf

TTestw_conf() liefert den aggregierten t-Wert zweier unabhängiger Stichproben.

TTestw_conf() liefert den aggregierten t-Wert zweier unabhängiger Stichproben. (weight, grp, value [, sig[, eq_var]]**)**

ttestw_df

TTestw_df() liefert den aggregierten df-Wert (Freiheitsgrade) für den Studenten-t-Test zweier unabhängiger Stichproben.

TTestw_df() liefert den aggregierten df-Wert (Freiheitsgrade) für den Studenten-t-Test zweier unabhängiger Stichproben. (weight, grp, value [, eq_var]**)**

ttestw_dif

TTestw_dif() liefert die aggregierte Mittelwertdifferenz für den Studenten-t-Test zweier unabhängiger Stichproben.

TTestw_dif() liefert die aggregierte Mittelwertdifferenz für den Studenten-t-Test zweier unabhängiger Stichproben. (weight, grp, value**)**

ttestw_lower

TTestw_lower() liefert den aggregierten Wert für das untere Ende des Konfidenzintervalls zweier unabhängiger Stichproben.

TTestw_lower() liefert den aggregierten Wert für das untere Ende des Konfidenzintervalls zweier unabhängiger Stichproben. (weight, grp, value [, sig[, eq_var]]**)**

ttestw_sig

TTestw_sig() liefert das 2-seitige Signifikanzniveau für den aggregierten Studenten-t-Test zweier unabhängiger Stichproben.

TTestw_sig() liefert das 2-seitige Signifikanzniveau für den aggregierten Studenten-t-Test zweier unabhängiger Stichproben. (weight, grp, value [, eq_var]**)**

ttestw_sterr

TTestw_sterr() liefert den Standardfehler der Mittelwertdifferenz für den aggregierten Studenten-t-Test zweier unabhängiger Stichproben.

TTestw_sterr() liefert den Standardfehler der Mittelwertdifferenz für den aggregierten Studenten-t-Test zweier unabhängiger Stichproben. (weight, grp, value [, eq_var]**)**

ttestw_t

TTestw_t() liefert den aggregierten t-Wert zweier unabhängiger Stichproben.

TTestw_t() liefert den aggregierten t-Wert zweier unabhängiger Stichproben. (weight, grp, value [, eq_var]**)**

ttestw_upper

TTestw_upper() liefert den aggregierten Wert für das obere Ende des Konfidenzintervalls zweier unabhängiger Stichproben.

TTestw_upper() liefert den aggregierten Wert für das obere Ende des Konfidenzintervalls zweier unabhängiger Stichproben. (weight, grp, value [, sig [, eq_var]]**)**

T-Tests mit einer Stichprobe

Die folgenden Funktionen beziehen sich auf den t-Test bei einer Stichprobe.

ttest1_conf

TTest1_conf() liefert das aggregierte Konfidenzintervall für eine Reihe von Stichproben.

TTest1_conf() liefert das aggregierte Konfidenzintervall für eine Reihe von Stichproben. (value [, sig]**)**

ttest1_df

TTest1_df() liefert den aggregierten df-Wert (Freiheitsgrade) für den Studenten-t-Test für eine Reihe von Stichproben.

TTest1_df() liefert den aggregierten df-Wert (Freiheitsgrade) für den Studenten-t-Test für eine Reihe von Stichproben. (value**)**

ttest1_dif

TTest1_dif() liefert die aggregierte Mittelwertdifferenz für den Studenten-t-Test einer Reihe von Stichproben.

TTest1_dif() liefert die aggregierte Mittelwertdifferenz für den Studenten-t-Test einer Reihe von Stichproben. (value**)**

ttest1_lower

TTest1_lower() liefert den aggregierten Wert für das untere Ende des Konfidenzintervalls für eine Reihe von Stichproben.

TTest1_lower() liefert den aggregierten Wert für das untere Ende des Konfidenzintervalls für eine Reihe von Stichproben. (value [, sig]**)**

ttest1_sig

TTest1_sig() liefert das aggregierte 2-seitige Signifikanzniveau für den Studenten-t-Test für eine Reihe von Stichproben.

TTest1_sig() liefert das aggregierte 2-seitige Signifikanzniveau für den Studenten-t-Test für eine Reihe von Stichproben. (value**)**

ttest1_sterr

TTest1_sterr() liefert den Standardfehler der Mittelwertdifferenz für den aggregierten Studenten-t-Test für eine Reihe von Stichproben.

TTest1_sterr() liefert den Standardfehler der Mittelwertdifferenz für den aggregierten Studenten-t-Test für eine Reihe von Stichproben. (value**)**

ttest1_t

TTest1_t() liefert den aggregierten t-Wert für eine Reihe von Stichproben.

TTest1_t() liefert den aggregierten t-Wert für eine Reihe von Stichproben. (value**)**

ttest1_upper

TTest1_upper() liefert den aggregierten Wert für das obere Ende des Konfidenzintervalls für eine Reihe von Stichproben.

TTest1_upper() liefert den aggregierten Wert für das obere Ende des Konfidenzintervalls für eine Reihe von Stichproben. (value [, sig]**)**

T-Test mit einer gewichteten Stichprobe

Die folgenden Funktionen beziehen sich auf den t-Test einer Stichprobe, bei dem die Werte der Stichprobe unterschiedlich gewichtet sind.

ttest1w_conf

TTest1w_conf() ist eine **numerische** Funktion, die das aggregierte Konfidenzintervall für eine Reihe von Stichproben liefert.

TTest1w_conf() ist eine numerische Funktion, die das aggregierte Konfidenzintervall für eine Reihe von Stichproben liefert. (weight, value [, sig]**)**

ttest1w_df

TTest1w_df() liefert den aggregierten df-Wert (Freiheitsgrade) für den Studenten-t-Test für eine Reihe von Stichproben.

TTest1w_df() liefert den aggregierten df-Wert (Freiheitsgrade) für den Studenten-t-Test für eine Reihe von Stichproben. (weight, value**)**

ttest1w_dif

TTest1w_dif() liefert die aggregierte Mittelwertdifferenz für den Studenten-t-Test einer Reihe von Stichproben.

TTest1w_dif() liefert die aggregierte Mittelwertdifferenz für den Studenten-t-Test einer Reihe von Stichproben. (weight, value**)**

ttest1w_lower

TTest1w_lower() liefert den aggregierten Wert für das untere Ende des Konfidenzintervalls für eine Reihe von Stichproben.

```
TTest1w_lower() liefert den aggregierten Wert für das untere Ende des
Konfidenzintervalls für eine Reihe von Stichproben. (weight, value [, sig])
```

ttest1w_sig

TTest1w_sig() liefert das aggregierte 2-seitige Signifikanzniveau für den Studenten-t-Test für eine Reihe von Stichproben.

```
TTest1w_sig() liefert das aggregierte 2-seitige Signifikanzniveau für den
Studenten-t-Test für eine Reihe von Stichproben. (weight, value)
```

ttest1w_sterr

TTest1w_sterr() liefert den Standardfehler der Mittelwertdifferenz für den aggregierten Studenten-t-Test für eine Reihe von Stichproben.

```
TTest1w_sterr() liefert den Standardfehler der Mittelwertdifferenz für den
aggregierten Studenten-t-Test für eine Reihe von Stichproben. (weight, value)
```

ttest1w_t

TTest1w_t() liefert den aggregierten t-Wert für eine Reihe von Stichproben.

```
TTest1w_t() liefert den aggregierten t-Wert für eine Reihe von Stichproben.
( weight, value)
```

ttest1w_upper

TTest1w_upper() liefert den aggregierten Wert für das obere Ende des Konfidenzintervalls für eine Reihe von Stichproben.

```
TTest1w_upper() liefert den aggregierten Wert für das obere Ende des
Konfidenzintervalls für eine Reihe von Stichproben. (weight, value [, sig])
```

TTest_conf

TTest_conf liefert den aggregierten t-Test-Wert des Konfidenzintervalls zweier unabhängiger Stichproben.

Die Funktion bezieht sich auf t-Tests für unabhängige Stichproben.

Falls die Funktion im Datenladeskript verwendet wird, werden die Werte über mehrere im group by-Zusatz bezeichnete Datensätze aggregiert.

Wird die Funktion als Diagrammformel verwendet, werden die Werte über die Diagrammdimensionen aggregiert.

Syntax:

```
TTest_conf ( grp, value [, sig [, eq_var]])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
value	Die zu interpretierenden Stichprobenwerte. Die Stichprobenwerte müssen logisch gruppiert werden, wie von exakt zwei Werten in **group** festgelegt. Wird im Ladeskript für den Stichprobenwert kein Feldname angegeben, erhält das Feld automatisch die Bezeichnung **Value**.
grp	Das Feld mit den Namen der beiden Stichprobengruppen. Wird im Ladeskript für die Gruppe kein Feldname angegeben, erhält das Feld automatisch die Bezeichnung **Type**.
sig	Das zweiseitige Signifikanzniveau wird durch **sig** angegeben. Ist der Parameter **sig** nicht definiert, wird er auf 0,025 gesetzt, was dem gängigen Konfidenzintervall von 95 % entspricht.
eq_var	Ergibt **eq_var** False (0), werden unterschiedliche Varianzen in beiden Stichproben angenommen. Ergibt **eq_var** True (1), werden gleiche Varianzen in beiden Stichproben angenommen.

Beschränkungen:

Bei Textwerten, NULL-Werten oder fehlenden Werten im Formelwert liefert die Funktion NULL.

Beispiele:

```
TTest_conf( Group, Value )
TTest_conf( Group, Value, Sig, false )
```

Siehe auch:

Erstellen eines typischen t-test-Reports (page 543)

TTest_df

TTest_df() liefert den aggregierten Wert (Freiheitsgrade) für den Studenten-t-Test zweier unabhängiger Stichproben.

Die Funktion bezieht sich auf t-Tests für unabhängige Stichproben.

Falls die Funktion im Datenladeskript verwendet wird, werden die Werte über mehrere im group by-Zusatz bezeichnete Datensätze aggregiert.

Wird die Funktion als Diagrammformel verwendet, werden die Werte über die Diagrammdimensionen aggregiert.

Syntax:

```
TTest_df (grp, value [, eq_var])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
value	Die zu interpretierenden Stichprobenwerte. Die Stichprobenwerte müssen logisch gruppiert werden, wie von exakt zwei Werten in **group** festgelegt. Wird im Ladeskript für den Stichprobenwert kein Feldname angegeben, erhält das Feld automatisch die Bezeichnung **Value**.
grp	Das Feld mit den Namen der beiden Stichprobengruppen. Wird im Ladeskript für die Gruppe kein Feldname angegeben, erhält das Feld automatisch die Bezeichnung **Type**.
eq_var	Ergibt **eq_var** False (0), werden unterschiedliche Varianzen in beiden Stichproben angenommen. Ergibt **eq_var** True (1), werden gleiche Varianzen in beiden Stichproben angenommen.

Beschränkungen:

Bei Textwerten, NULL-Werten oder fehlenden Werten im Formelwert liefert die Funktion NULL.

Beispiele:

```
TTest_df( Group, Value )
TTest_df( Group, Value, false )
```

Siehe auch:

Erstellen eines typischen t-test-Reports (page 543)

TTest_dif

TTest_dif() ist eine numerische Funktion, die die aggregierte Mittelwertdifferenz für den Studenten-t-Test zweier unabhängiger Stichproben liefert.

Die Funktion bezieht sich auf t-Tests für unabhängige Stichproben.

Falls die Funktion im Datenladeskript verwendet wird, werden die Werte über mehrere im group by-Zusatz bezeichnete Datensätze aggregiert.

Wird die Funktion als Diagrammformel verwendet, werden die Werte über die Diagrammdimensionen aggregiert.

Syntax:

```
TTest_dif (grp, value [, eq_var] )
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
value	Die zu interpretierenden Stichprobenwerte. Die Stichprobenwerte müssen logisch gruppiert werden, wie von exakt zwei Werten in **group** festgelegt. Wird im Ladeskript für den Stichprobenwert kein Feldname angegeben, erhält das Feld automatisch die Bezeichnung **Value**.
grp	Das Feld mit den Namen der beiden Stichprobengruppen. Wird im Ladeskript für die Gruppe kein Feldname angegeben, erhält das Feld automatisch die Bezeichnung **Type**.
eq_var	Ergibt **eq_var** False (0), werden unterschiedliche Varianzen in beiden Stichproben angenommen. Ergibt **eq_var** True (1), werden gleiche Varianzen in beiden Stichproben angenommen.

Beschränkungen:

Bei Textwerten, NULL-Werten oder fehlenden Werten im Formelwert liefert die Funktion NULL.

Beispiele:

```
TTest_dif( Group, Value )
TTest_dif( Group, Value, false )
```

Siehe auch:

Erstellen eines typischen t-test-Reports (page 543)

TTest_lower

TTest_lower() liefert den aggregierten Wert für das untere Ende des Konfidenzintervalls zweier unabhängiger Stichproben.

Die Funktion bezieht sich auf t-Tests für unabhängige Stichproben.

Falls die Funktion im Datenladeskript verwendet wird, werden die Werte über mehrere im group by-Zusatz bezeichnete Datensätze aggregiert.

Wird die Funktion als Diagrammformel verwendet, werden die Werte über die Diagrammdimensionen aggregiert.

Syntax:

```
TTest_lower (grp, value [, sig [, eq_var]])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
value	Die zu interpretierenden Stichprobenwerte. Die Stichprobenwerte müssen logisch gruppiert werden, wie von exakt zwei Werten in **group** festgelegt. Wird im Ladeskript für den Stichprobenwert kein Feldname angegeben, erhält das Feld automatisch die Bezeichnung **Value**.
grp	Das Feld mit den Namen der beiden Stichprobengruppen. Wird im Ladeskript für die Gruppe kein Feldname angegeben, erhält das Feld automatisch die Bezeichnung **Type**.
sig	Das zweiseitige Signifikanzniveau wird durch **sig** angegeben. Ist der Parameter **sig** nicht definiert, wird er auf 0,025 gesetzt, was dem gängigen Konfidenzintervall von 95 % entspricht.
eq_var	Ergibt **eq_var** False (0), werden unterschiedliche Varianzen in beiden Stichproben angenommen. Ergibt **eq_var** True (1), werden gleiche Varianzen in beiden Stichproben angenommen.

Beschränkungen:

Bei Textwerten, NULL-Werten oder fehlenden Werten im Formelwert liefert die Funktion NULL.

Beispiele:

```
TTest_lower( Group, Value )
TTest_lower( Group, Value, Sig, false )
```

Siehe auch:

Erstellen eines typischen t-test-Reports (page 543)

TTest_sig

TTest_sig() liefert das 2-seitige Signifikanzniveau für den aggregierten Studenten-t-Test zweier unabhängiger Stichproben.

Die Funktion bezieht sich auf t-Tests für unabhängige Stichproben.

Falls die Funktion im Datenladeskript verwendet wird, werden die Werte über mehrere im group by-Zusatz bezeichnete Datensätze aggregiert.

Wird die Funktion als Diagrammformel verwendet, werden die Werte über die Diagrammdimensionen aggregiert.

Syntax:

```
TTest_sig (grp, value [, eq_var])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
value	Die zu interpretierenden Stichprobenwerte. Die Stichprobenwerte müssen logisch gruppiert werden, wie von exakt zwei Werten in **group** festgelegt. Wird im Ladeskript für den Stichprobenwert kein Feldname angegeben, erhält das Feld automatisch die Bezeichnung **Value**.
grp	Das Feld mit den Namen der beiden Stichprobengruppen. Wird im Ladeskript für die Gruppe kein Feldname angegeben, erhält das Feld automatisch die Bezeichnung **Type**.
eq_var	Ergibt **eq_var** False (0), werden unterschiedliche Varianzen in beiden Stichproben angenommen. Ergibt **eq_var** True (1), werden gleiche Varianzen in beiden Stichproben angenommen.

Beschränkungen:

Bei Textwerten, NULL-Werten oder fehlenden Werten im Formelwert liefert die Funktion NULL.

Beispiele:

```
TTest_sig( Group, Value )
TTest_sig( Group, Value, false )
```

Siehe auch:

Erstellen eines typischen t-test-Reports (page 543)

TTest_sterr

TTest_sterr() liefert den Standardfehler der Mittelwertdifferenz für den aggregierten Studenten-t-Test zweier unabhängiger Stichproben.

Die Funktion bezieht sich auf t-Tests für unabhängige Stichproben.

Falls die Funktion im Datenladeskript verwendet wird, werden die Werte über mehrere im group by-Zusatz bezeichnete Datensätze aggregiert.

Wird die Funktion als Diagrammformel verwendet, werden die Werte über die Diagrammdimensionen aggregiert.

Syntax:

```
TTest_sterr (grp, value [, eq_var])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
value	Die zu interpretierenden Stichprobenwerte. Die Stichprobenwerte müssen logisch gruppiert werden, wie von exakt zwei Werten in **group** festgelegt. Wird im Ladeskript für den Stichprobenwert kein Feldname angegeben, erhält das Feld automatisch die Bezeichnung **Value**.
grp	Das Feld mit den Namen der beiden Stichprobengruppen. Wird im Ladeskript für die Gruppe kein Feldname angegeben, erhält das Feld automatisch die Bezeichnung **Type**.
eq_var	Ergibt **eq_var** False (0), werden unterschiedliche Varianzen in beiden Stichproben angenommen. Ergibt **eq_var** True (1), werden gleiche Varianzen in beiden Stichproben angenommen.

Beschränkungen:

Bei Textwerten, NULL-Werten oder fehlenden Werten im Formelwert liefert die Funktion NULL.

Beispiele:

```
TTest_sterr( Group, Value )
TTest_sterr( Group, Value, false )
```

Siehe auch:

Erstellen eines typischen t-test-Reports (page 543)

TTest_t

TTest_t() liefert den aggregierten t-Wert zweier unabhängiger Stichproben.

Die Funktion bezieht sich auf t-Tests für unabhängige Stichproben.

Falls die Funktion im Datenladeskript verwendet wird, werden die Werte über mehrere im group by-Zusatz bezeichnete Datensätze aggregiert.

Wird die Funktion als Diagrammformel verwendet, werden die Werte über die Diagrammdimensionen aggregiert.

Syntax:

```
TTest_t(grp, value[, eq_var])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
value	Die zu interpretierenden Stichprobenwerte. Die Stichprobenwerte müssen logisch gruppiert werden, wie von exakt zwei Werten in **group** festgelegt. Wird im Ladeskript für den Stichprobenwert kein Feldname angegeben, erhält das Feld automatisch die Bezeichnung **Value**.
grp	Das Feld mit den Namen der beiden Stichprobengruppen. Wird im Ladeskript für die Gruppe kein Feldname angegeben, erhält das Feld automatisch die Bezeichnung **Type**.
eq_var	Ergibt **eq_var** False (0), werden unterschiedliche Varianzen in beiden Stichproben angenommen. Ergibt **eq_var** True (1), werden gleiche Varianzen in beiden Stichproben angenommen.

Beschränkungen:

Bei Textwerten, NULL-Werten oder fehlenden Werten im Formelwert liefert die Funktion NULL.

Beispiel:

```
TTest_t( Group, Value, false )
```

Siehe auch:

Erstellen eines typischen t-test-Reports (page 543)

TTest_upper

TTest_upper() liefert den aggregierten Wert für das obere Ende des Konfidenzintervalls zweier unabhängiger Stichproben.

Die Funktion bezieht sich auf t-Tests für unabhängige Stichproben.

Falls die Funktion im Datenladeskript verwendet wird, werden die Werte über mehrere im group by-Zusatz bezeichnete Datensätze aggregiert.

Wird die Funktion als Diagrammformel verwendet, werden die Werte über die Diagrammdimensionen aggregiert.

Syntax:

```
TTest_upper (grp, value [, sig [, eq_var]])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
value	Die zu interpretierenden Stichprobenwerte. Die Stichprobenwerte müssen logisch gruppiert werden, wie von exakt zwei Werten in **group** festgelegt. Wird im Ladeskript für den Stichprobenwert kein Feldname angegeben, erhält das Feld automatisch die Bezeichnung **Value**.
grp	Das Feld mit den Namen der beiden Stichprobengruppen. Wird im Ladeskript für die Gruppe kein Feldname angegeben, erhält das Feld automatisch die Bezeichnung **Type**.
sig	Das zweiseitige Signifikanzniveau wird durch **sig** angegeben. Ist der Parameter **sig** nicht definiert, wird er auf 0,025 gesetzt, was dem gängigen Konfidenzintervall von 95 % entspricht.
eq_var	Ergibt **eq_var** False (0), werden unterschiedliche Varianzen in beiden Stichproben angenommen. Ergibt **eq_var** True (1), werden gleiche Varianzen in beiden Stichproben angenommen.

Beschränkungen:

Bei Textwerten, NULL-Werten oder fehlenden Werten im Formelwert liefert die Funktion NULL.

Beispiele:

```
TTest_upper( Group, Value )
TTest_upper( Group, Value, sig, false )
```

Siehe auch:

Erstellen eines typischen t-test-Reports (page 543)

TTestw_conf

TTestw_conf() liefert den aggregierten t-Wert zweier unabhängiger Stichproben.

Diese Funktion bezieht sich auf t-Tests von zwei unabhängigen Stichproben, bei denen die Werte der Stichprobe unterschiedlich gewichtet sind.

Falls die Funktion im Datenladeskript verwendet wird, werden die Werte über mehrere im group by-Zusatz bezeichnete Datensätze aggregiert.

Wird die Funktion als Diagrammformel verwendet, werden die Werte über die Diagrammdimensionen aggregiert.

Syntax:

```
TTestw_conf (weight, grp, value [, sig [, eq_var]])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
value	Die zu interpretierenden Stichprobenwerte. Die Stichprobenwerte müssen logisch gruppiert werden, wie von exakt zwei Werten in **group** festgelegt. Wird im Ladeskript für den Stichprobenwert kein Feldname angegeben, erhält das Feld automatisch die Bezeichnung **Value**.
weight	Jeder Wert in **value** geht entsprechend der zugehörigen Gewichtung einmal oder mehrfach in die Berechnung **weight** ein.
grp	Das Feld mit den Namen der beiden Stichprobengruppen. Wird im Ladeskript für die Gruppe kein Feldname angegeben, erhält das Feld automatisch die Bezeichnung **Type**.
sig	Das zweiseitige Signifikanzniveau wird durch **sig** angegeben. Ist der Parameter **sig** nicht definiert, wird er auf 0,025 gesetzt, was dem gängigen Konfidenzintervall von 95 % entspricht.
eq_var	Ergibt **eq_var** False (0), werden unterschiedliche Varianzen in beiden Stichproben angenommen. Ergibt **eq_var** True (1), werden gleiche Varianzen in beiden Stichproben angenommen.

Beschränkungen:

Bei Textwerten, NULL-Werten oder fehlenden Werten im Formelwert liefert die Funktion NULL.

Beispiele:

```
TTestw_conf( Weight, Group, Value )
TTestw_conf( Weight, Group, Value, sig, false )
```

Siehe auch:

Erstellen eines typischen t-test-Reports (page 543)

TTestw_df

TTestw_df() liefert den aggregierten df-Wert (Freiheitsgrade) für den Studenten-t-Test zweier unabhängiger Stichproben.

Diese Funktion bezieht sich auf t-Tests von zwei unabhängigen Stichproben, bei denen die Werte der Stichprobe unterschiedlich gewichtet sind.

Falls die Funktion im Datenladeskript verwendet wird, werden die Werte über mehrere im group by-Zusatz bezeichnete Datensätze aggregiert.

Wird die Funktion als Diagrammformel verwendet, werden die Werte über die Diagrammdimensionen aggregiert.

Syntax:

```
TTestw_df (weight, grp, value [, eq_var])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
weight	Jeder Wert in **value** geht entsprechend der zugehörigen Gewichtung einmal oder mehrfach in die Berechnung **weight** ein.
grp	Das Feld mit den Namen der beiden Stichprobengruppen. Wird im Ladeskript für die Gruppe kein Feldname angegeben, erhält das Feld automatisch die Bezeichnung **Type**.
value	Die zu interpretierenden Stichprobenwerte. Die Stichprobenwerte müssen logisch gruppiert werden, wie von exakt zwei Werten in **group** festgelegt. Wird im Ladeskript für den Stichprobenwert kein Feldname angegeben, erhält das Feld automatisch die Bezeichnung **Value**.
eq_var	Ergibt **eq_var** False (0), werden unterschiedliche Varianzen in beiden Stichproben angenommen. Ergibt **eq_var** True (1), werden gleiche Varianzen in beiden Stichproben angenommen.

Beschränkungen:

Bei Textwerten, NULL-Werten oder fehlenden Werten im Formelwert liefert die Funktion NULL.

Beispiele:

```
TTestw_df( Weight, Group, Value )
TTestw_df( Weight, Group, Value, false )
```

Siehe auch:

Erstellen eines typischen t-test-Reports (page 543)

TTestw_dif

TTestw_dif() liefert die aggregierte Mittelwertdifferenz für den Studenten-t-Test zweier unabhängiger Stichproben.

Diese Funktion bezieht sich auf t-Tests von zwei unabhängigen Stichproben, bei denen die Werte der Stichprobe unterschiedlich gewichtet sind.

Falls die Funktion im Datenladeskript verwendet wird, werden die Werte über mehrere im group by-Zusatz bezeichnete Datensätze aggregiert.

Wird die Funktion als Diagrammformel verwendet, werden die Werte über die Diagrammdimensionen aggregiert.

Syntax:

```
TTestw_dif (weight, grp, value)
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
weight	Jeder Wert in **value** geht entsprechend der zugehörigen Gewichtung einmal oder mehrfach in die Berechnung **weight** ein.
grp	Das Feld mit den Namen der beiden Stichprobengruppen. Wird im Ladeskript für die Gruppe kein Feldname angegeben, erhält das Feld automatisch die Bezeichnung **Type**.
value	Die zu interpretierenden Stichprobenwerte. Die Stichprobenwerte müssen logisch gruppiert werden, wie von exakt zwei Werten in **group** festgelegt. Wird im Ladeskript für den Stichprobenwert kein Feldname angegeben, erhält das Feld automatisch die Bezeichnung **Value**.

Beschränkungen:

Bei Textwerten, NULL-Werten oder fehlenden Werten im Formelwert liefert die Funktion NULL.

Beispiele:

```
TTestw_dif( Weight, Group, Value )
TTestw_dif( Weight, Group, Value, false )
```

Siehe auch:

Erstellen eines typischen t-test-Reports (page 543)

TTestw_lower

TTestw_lower() liefert den aggregierten Wert für das untere Ende des Konfidenzintervalls zweier unabhängiger Stichproben.

Diese Funktion bezieht sich auf t-Tests von zwei unabhängigen Stichproben, bei denen die Werte der Stichprobe unterschiedlich gewichtet sind.

Falls die Funktion im Datenladeskript verwendet wird, werden die Werte über mehrere im group by-Zusatz bezeichnete Datensätze aggregiert.

Wird die Funktion als Diagrammformel verwendet, werden die Werte über die Diagrammdimensionen aggregiert.

Syntax:

```
TTestw_lower (weight, grp, value [, sig [, eq_var]])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
weight	Jeder Wert in **value** geht entsprechend der zugehörigen Gewichtung einmal oder mehrfach in die Berechnung **weight** ein.
grp	Das Feld mit den Namen der beiden Stichprobengruppen. Wird im Ladeskript für die Gruppe kein Feldname angegeben, erhält das Feld automatisch die Bezeichnung **Type**.
value	Die zu interpretierenden Stichprobenwerte. Die Stichprobenwerte müssen logisch gruppiert werden, wie von exakt zwei Werten in **group** festgelegt. Wird im Ladeskript für den Stichprobenwert kein Feldname angegeben, erhält das Feld automatisch die Bezeichnung **Value**.
sig	Das zweiseitige Signifikanzniveau wird durch **sig** angegeben. Ist der Parameter **sig** nicht definiert, wird er auf 0,025 gesetzt, was dem gängigen Konfidenzintervall von 95 % entspricht.
eq_var	Ergibt **eq_var** False (0), werden unterschiedliche Varianzen in beiden Stichproben angenommen. Ergibt **eq_var** True (1), werden gleiche Varianzen in beiden Stichproben angenommen.

Beschränkungen:

Bei Textwerten, NULL-Werten oder fehlenden Werten im Formelwert liefert die Funktion NULL.

Beispiele:

```
TTestw_lower( Weight, Group, Value )
TTestw_lower( Weight, Group, Value, sig, false )
```

Siehe auch:

Erstellen eines typischen t-test-Reports (page 543)

TTestw_sig

TTestw_sig() liefert das 2-seitige Signifikanzniveau für den aggregierten Studenten-t-Test zweier unabhängiger Stichproben.

Diese Funktion bezieht sich auf t-Tests von zwei unabhängigen Stichproben, bei denen die Werte der Stichprobe unterschiedlich gewichtet sind.

Falls die Funktion im Datenladeskript verwendet wird, werden die Werte über mehrere im group by-Zusatz bezeichnete Datensätze aggregiert.

Wird die Funktion als Diagrammformel verwendet, werden die Werte über die Diagrammdimensionen aggregiert.

Syntax:

```
TTestw_sig ( weight, grp, value [, eq_var])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
weight	Jeder Wert in **value** geht entsprechend der zugehörigen Gewichtung einmal oder mehrfach in die Berechnung **weight** ein.
grp	Das Feld mit den Namen der beiden Stichprobengruppen. Wird im Ladeskript für die Gruppe kein Feldname angegeben, erhält das Feld automatisch die Bezeichnung **Type**.
value	Die zu interpretierenden Stichprobenwerte. Die Stichprobenwerte müssen logisch gruppiert werden, wie von exakt zwei Werten in **group** festgelegt. Wird im Ladeskript für den Stichprobenwert kein Feldname angegeben, erhält das Feld automatisch die Bezeichnung **Value**.
eq_var	Ergibt **eq_var** False (0), werden unterschiedliche Varianzen in beiden Stichproben angenommen. Ergibt **eq_var** True (1), werden gleiche Varianzen in beiden Stichproben angenommen.

Beschränkungen:

Bei Textwerten, NULL-Werten oder fehlenden Werten im Formelwert liefert die Funktion NULL.

Beispiele:

```
TTestw_sig( Weight, Group, Value )
TTestw_sig( Weight, Group, Value, false )
```

Siehe auch:

Erstellen eines typischen t-test-Reports (page 543)

TTestw_sterr

TTestw_sterr() liefert den Standardfehler der Mittelwertdifferenz für den aggregierten Studenten-t-Test zweier unabhängiger Stichproben.

Diese Funktion bezieht sich auf t-Tests von zwei unabhängigen Stichproben, bei denen die Werte der Stichprobe unterschiedlich gewichtet sind.

Falls die Funktion im Datenladeskript verwendet wird, werden die Werte über mehrere im group by-Zusatz bezeichnete Datensätze aggregiert.

Wird die Funktion als Diagrammformel verwendet, werden die Werte über die Diagrammdimensionen aggregiert.

Syntax:

```
TTestw_sterr (weight, grp, value [, eq_var])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
weight	Jeder Wert in **value** geht entsprechend der zugehörigen Gewichtung einmal oder mehrfach in die Berechnung **weight** ein.
grp	Das Feld mit den Namen der beiden Stichprobengruppen. Wird im Ladeskript für die Gruppe kein Feldname angegeben, erhält das Feld automatisch die Bezeichnung **Type**.
value	Die zu interpretierenden Stichprobenwerte. Die Stichprobenwerte müssen logisch gruppiert werden, wie von exakt zwei Werten in **group** festgelegt. Wird im Ladeskript für den Stichprobenwert kein Feldname angegeben, erhält das Feld automatisch die Bezeichnung **Value**.
eq_var	Ergibt **eq_var** False (0), werden unterschiedliche Varianzen in beiden Stichproben angenommen. Ergibt **eq_var** True (1), werden gleiche Varianzen in beiden Stichproben angenommen.

Beschränkungen:

Bei Textwerten, NULL-Werten oder fehlenden Werten im Formelwert liefert die Funktion NULL.

Beispiele:

```
TTestw_sterr( Weight, Group, Value )
TTestw_sterr( Weight, Group, Value, false )
```

Siehe auch:

Erstellen eines typischen t-test-Reports (page 543)

TTestw_t

TTestw_t() liefert den aggregierten t-Wert zweier unabhängiger Stichproben.

Diese Funktion bezieht sich auf t-Tests von zwei unabhängigen Stichproben, bei denen die Werte der Stichprobe unterschiedlich gewichtet sind.

Falls die Funktion im Datenladeskript verwendet wird, werden die Werte über mehrere im group by-Zusatz bezeichnete Datensätze aggregiert.

Wird die Funktion als Diagrammformel verwendet, werden die Werte über die Diagrammdimensionen aggregiert.

Syntax:

```
ttestw_t (weight, grp, value [, eq_var])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
value	Die zu interpretierenden Stichprobenwerte. Die Stichprobenwerte müssen logisch gruppiert werden, wie von exakt zwei Werten in **group** festgelegt. Wird im Ladeskript für den Stichprobenwert kein Feldname angegeben, erhält das Feld automatisch die Bezeichnung **Value**.
weight	Jeder Wert in **value** geht entsprechend der zugehörigen Gewichtung einmal oder mehrfach in die Berechnung **weight** ein.
grp	Das Feld mit den Namen der beiden Stichprobengruppen. Wird im Ladeskript für die Gruppe kein Feldname angegeben, erhält das Feld automatisch die Bezeichnung **Type**.
eq_var	Ergibt **eq_var** False (0), werden unterschiedliche Varianzen in beiden Stichproben angenommen. Ergibt **eq_var** True (1), werden gleiche Varianzen in beiden Stichproben angenommen.

Beschränkungen:

Bei Textwerten, NULL-Werten oder fehlenden Werten im Formelwert liefert die Funktion NULL.

Beispiele:

```
TTestw_t( Weight, Group, Value )
TTestw_t( Weight, Group, Value, false )
```

Siehe auch:

Erstellen eines typischen t-test-Reports (page 543)

TTestw_upper

TTestw_upper() liefert den aggregierten Wert für das obere Ende des Konfidenzintervalls zweier unabhängiger Stichproben.

Diese Funktion bezieht sich auf t-Tests von zwei unabhängigen Stichproben, bei denen die Werte der Stichprobe unterschiedlich gewichtet sind.

Falls die Funktion im Datenladeskript verwendet wird, werden die Werte über mehrere im group by-Zusatz bezeichnete Datensätze aggregiert.

Wird die Funktion als Diagrammformel verwendet, werden die Werte über die Diagrammdimensionen aggregiert.

Syntax:

```
TTestw_upper (weight, grp, value [, sig [, eq_var]])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
weight	Jeder Wert in **value** geht entsprechend der zugehörigen Gewichtung einmal oder mehrfach in die Berechnung **weight** ein.
grp	Das Feld mit den Namen der beiden Stichprobengruppen. Wird im Ladeskript für die Gruppe kein Feldname angegeben, erhält das Feld automatisch die Bezeichnung **Type**.
value	Die zu interpretierenden Stichprobenwerte. Die Stichprobenwerte müssen logisch gruppiert werden, wie von exakt zwei Werten in **group** festgelegt. Wird im Ladeskript für den Stichprobenwert kein Feldname angegeben, erhält das Feld automatisch die Bezeichnung **Value**.
sig	Das zweiseitige Signifikanzniveau wird durch **sig** angegeben. Ist der Parameter **sig** nicht definiert, wird er auf 0,025 gesetzt, was dem gängigen Konfidenzintervall von 95 % entspricht.
eq_var	Ergibt **eq_var** False (0), werden unterschiedliche Varianzen in beiden Stichproben angenommen. Ergibt **eq_var** True (1), werden gleiche Varianzen in beiden Stichproben angenommen.

Beschränkungen:

Bei Textwerten, NULL-Werten oder fehlenden Werten im Formelwert liefert die Funktion NULL.

Beispiele:

```
TTestw_upper( Weight, Group, Value )
TTestw_upper( Weight, Group, Value, sig, false )
```

Siehe auch:

Erstellen eines typischen t-test-Reports (page 543)

TTest1_conf

TTest1_conf() liefert das aggregierte Konfidenzintervall für eine Reihe von Stichproben.

Die Funktion bezieht sich auf t-Tests einer Stichprobe.

Falls die Funktion im Datenladeskript verwendet wird, werden die Werte über mehrere im group by-Zusatz bezeichnete Datensätze aggregiert.

Wird die Funktion als Diagrammformel verwendet, werden die Werte über die Diagrammdimensionen aggregiert.

Syntax:

```
TTest1_conf (value [, sig ])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
value	Die auszuwertenden Stichproben. Wird im Ladeskript für den Stichprobenwert kein Feldname angegeben, erhält das Feld automatisch die Bezeichnung **Value**.
sig	Das zweiseitige Signifikanzniveau wird durch **sig** angegeben. Ist der Parameter **sig** nicht definiert, wird er auf 0,025 gesetzt, was dem gängigen Konfidenzintervall von 95 % entspricht.

Beschränkungen:

Bei Textwerten, NULL-Werten oder fehlenden Werten im Formelwert liefert die Funktion NULL.

Beispiele:

```
TTest1_conf( Value )
TTest1_conf( Value, 0.005 )
```

Siehe auch:

Erstellen eines typischen t-test-Reports (page 543)

TTest1_df

TTest1_df() liefert den aggregierten df-Wert (Freiheitsgrade) für den Studenten-t-Test für eine Reihe von Stichproben.

Die Funktion bezieht sich auf t-Tests einer Stichprobe.

Falls die Funktion im Datenladeskript verwendet wird, werden die Werte über mehrere im group by-Zusatz bezeichnete Datensätze aggregiert.

Wird die Funktion als Diagrammformel verwendet, werden die Werte über die Diagrammdimensionen aggregiert.

Syntax:

```
TTest1_df (value)
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
value	Die auszuwertenden Stichproben. Wird im Ladeskript für den Stichprobenwert kein Feldname angegeben, erhält das Feld automatisch die Bezeichnung **Value**.

Beschränkungen:

Bei Textwerten, NULL-Werten oder fehlenden Werten im Formelwert liefert die Funktion NULL.

Beispiel:

```
TTest1_df( value )
```

Siehe auch:

Erstellen eines typischen t-test-Reports (page 543)

TTest1_dif

TTest1_dif() liefert die aggregierte Mittelwertdifferenz für den Studenten-t-Test einer Reihe von Stichproben.

Die Funktion bezieht sich auf t-Tests einer Stichprobe.

Falls die Funktion im Datenladeskript verwendet wird, werden die Werte über mehrere im group by-Zusatz bezeichnete Datensätze aggregiert.

Wird die Funktion als Diagrammformel verwendet, werden die Werte über die Diagrammdimensionen aggregiert.

Syntax:

```
TTest1_dif (value)
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
value	Die auszuwertenden Stichproben. Wird im Ladeskript für den Stichprobenwert kein Feldname angegeben, erhält das Feld automatisch die Bezeichnung **Value**.

Beschränkungen:

Bei Textwerten, NULL-Werten oder fehlenden Werten im Formelwert liefert die Funktion NULL.

Beispiel:

```
TTest1_dif( value )
```

Siehe auch:

Erstellen eines typischen t-test-Reports (page 543)

TTest1_lower

TTest1_lower() liefert den aggregierten Wert für das untere Ende des Konfidenzintervalls für eine Reihe von Stichproben.

Die Funktion bezieht sich auf t-Tests einer Stichprobe.

Falls die Funktion im Datenladeskript verwendet wird, werden die Werte über mehrere im group by-Zusatz bezeichnete Datensätze aggregiert.

Wird die Funktion als Diagrammformel verwendet, werden die Werte über die Diagrammdimensionen aggregiert.

Syntax:

```
TTest1_lower (value [, sig])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
value	Die auszuwertenden Stichproben. Wird im Ladeskript für den Stichprobenwert kein Feldname angegeben, erhält das Feld automatisch die Bezeichnung **Value**.
sig	Das zweiseitige Signifikanzniveau wird durch **sig** angegeben. Ist der Parameter **sig** nicht definiert, wird er auf 0,025 gesetzt, was dem gängigen Konfidenzintervall von 95 % entspricht.

Beschränkungen:

Bei Textwerten, NULL-Werten oder fehlenden Werten im Formelwert liefert die Funktion NULL.

Beispiele:

```
TTest1_lower( Value )
TTest1_lower( Value, 0.005 )
```

Siehe auch:

Erstellen eines typischen t-test-Reports (page 543)

TTest1_sig

TTest1_sig() liefert das aggregierte 2-seitige Signifikanzniveau für den Studenten-t-Test für eine Reihe von Stichproben.

Die Funktion bezieht sich auf t-Tests einer Stichprobe.

Falls die Funktion im Datenladeskript verwendet wird, werden die Werte über mehrere im group by-Zusatz bezeichnete Datensätze aggregiert.

Wird die Funktion als Diagrammformel verwendet, werden die Werte über die Diagrammdimensionen aggregiert.

Syntax:

```
TTest1_sig (value)
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
value	Die auszuwertenden Stichproben. Wird im Ladeskript für den Stichprobenwert kein Feldname angegeben, erhält das Feld automatisch die Bezeichnung **Value**.

Beschränkungen:

Bei Textwerten, NULL-Werten oder fehlenden Werten im Formelwert liefert die Funktion NULL.

Beispiel:

```
TTest1_sig( Value )
```

Siehe auch:

Erstellen eines typischen t-test-Reports (page 543)

TTest1_sterr

TTest1_sterr() liefert den Standardfehler der Mittelwertdifferenz für den aggregierten Studenten-t-Test für eine Reihe von Stichproben.

Die Funktion bezieht sich auf t-Tests einer Stichprobe.

Falls die Funktion im Datenladeskript verwendet wird, werden die Werte über mehrere im group by-Zusatz bezeichnete Datensätze aggregiert.

Wird die Funktion als Diagrammformel verwendet, werden die Werte über die Diagrammdimensionen aggregiert.

Syntax:

```
TTest1_sterr (value)
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
value	Die auszuwertenden Stichproben. Wird im Ladeskript für den Stichprobenwert kein Feldname angegeben, erhält das Feld automatisch die Bezeichnung **Value**.

Beschränkungen:

Bei Textwerten, NULL-Werten oder fehlenden Werten im Formelwert liefert die Funktion NULL.

Beispiel:

```
TTest1_sterr( Value )
```

Siehe auch:

Erstellen eines typischen t-test-Reports (page 543)

TTest1_t

TTest1_t() liefert den aggregierten t-Wert für eine Reihe von Stichproben.

Die Funktion bezieht sich auf t-Tests einer Stichprobe.

Falls die Funktion im Datenladeskript verwendet wird, werden die Werte über mehrere im group by-Zusatz bezeichnete Datensätze aggregiert.

Wird die Funktion als Diagrammformel verwendet, werden die Werte über die Diagrammdimensionen aggregiert.

Syntax:

```
TTest1_t (value)
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
value	Die auszuwertenden Stichproben. Wird im Ladeskript für den Stichprobenwert kein Feldname angegeben, erhält das Feld automatisch die Bezeichnung **Value**.

Beschränkungen:

Bei Textwerten, NULL-Werten oder fehlenden Werten im Formelwert liefert die Funktion NULL.

Beispiel:

```
TTest1_t( Value )
```

Siehe auch:

Erstellen eines typischen t-test-Reports (page 543)

TTest1_upper

TTest1_upper() liefert den aggregierten Wert für das obere Ende des Konfidenzintervalls für eine Reihe von Stichproben.

Die Funktion bezieht sich auf t-Tests einer Stichprobe.

Falls die Funktion im Datenladeskript verwendet wird, werden die Werte über mehrere im group by-Zusatz bezeichnete Datensätze aggregiert.

Wird die Funktion als Diagrammformel verwendet, werden die Werte über die Diagrammdimensionen aggregiert.

Syntax:

```
TTest1_upper (value [, sig])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
value	Die auszuwertenden Stichproben. Wird im Ladeskript für den Stichprobenwert kein Feldname angegeben, erhält das Feld automatisch die Bezeichnung **Value**.

Argument	Beschreibung
sig	Das zweiseitige Signifikanzniveau wird durch **sig** angegeben. Ist der Parameter **sig** nicht definiert, wird er auf 0,025 gesetzt, was dem gängigen Konfidenzintervall von 95 % entspricht.

Beschränkungen:

Bei Textwerten, NULL-Werten oder fehlenden Werten im Formelwert liefert die Funktion NULL.

Beispiele:

```
TTest1_upper( value )
TTest1_upper( value, 0.005 )
```

Siehe auch:

Erstellen eines typischen t-test-Reports (page 543)

TTest1w_conf

TTest1w_conf() ist eine **numerische** Funktion, die das aggregierte Konfidenzintervall für eine Reihe von Stichproben liefert.

Diese Funktion bezieht sich auf t-Tests einer Stichprobe, bei denen die Werte der Stichprobe unterschiedlich gewichtet sind.

Falls die Funktion im Datenladeskript verwendet wird, werden die Werte über mehrere im group by-Zusatz bezeichnete Datensätze aggregiert.

Wird die Funktion als Diagrammformel verwendet, werden die Werte über die Diagrammdimensionen aggregiert.

Syntax:

```
TTest1w_conf (weight, value [, sig ])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
value	Die auszuwertenden Stichproben. Wird im Ladeskript für den Stichprobenwert kein Feldname angegeben, erhält das Feld automatisch die Bezeichnung **Value**.
weight	Jeder Wert in **value** geht entsprechend der zugehörigen Gewichtung einmal oder mehrfach in die Berechnung **weight** ein.

Argument	Beschreibung
sig	Das zweiseitige Signifikanzniveau wird durch **sig** angegeben. Ist der Parameter **sig** nicht definiert, wird er auf 0,025 gesetzt, was dem gängigen Konfidenzintervall von 95 % entspricht.

Beschränkungen:

Bei Textwerten, NULL-Werten oder fehlenden Werten im Formelwert liefert die Funktion NULL.

Beispiele:

```
TTest1w_conf( Weight, Value )
TTest1w_conf( Weight, Value, 0.005 )
```

Siehe auch:

Erstellen eines typischen t-test-Reports (page 543)

TTest1w_df

TTest1w_df() liefert den aggregierten df-Wert (Freiheitsgrade) für den Studenten-t-Test für eine Reihe von Stichproben.

Diese Funktion bezieht sich auf t-Tests einer Stichprobe, bei denen die Werte der Stichprobe unterschiedlich gewichtet sind.

Falls die Funktion im Datenladeskript verwendet wird, werden die Werte über mehrere im group by-Zusatz bezeichnete Datensätze aggregiert.

Wird die Funktion als Diagrammformel verwendet, werden die Werte über die Diagrammdimensionen aggregiert.

Syntax:

```
TTest1w_df (weight, value)
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
value	Die auszuwertenden Stichproben. Wird im Ladeskript für den Stichprobenwert kein Feldname angegeben, erhält das Feld automatisch die Bezeichnung **Value**.
weight	Jeder Wert in **value** geht entsprechend der zugehörigen Gewichtung einmal oder mehrfach in die Berechnung **weight** ein.

Beschränkungen:

Bei Textwerten, NULL-Werten oder fehlenden Werten im Formelwert liefert die Funktion NULL.

Beispiel:

```
TTest1w_df( Weight, Value )
```

Siehe auch:

Erstellen eines typischen t-test-Reports (page 543)

TTest1w_dif

TTest1w_dif() liefert die aggregierte Mittelwertdifferenz für den Studenten-t-Test einer Reihe von Stichproben.

Diese Funktion bezieht sich auf t-Tests einer Stichprobe, bei denen die Werte der Stichprobe unterschiedlich gewichtet sind.

Falls die Funktion im Datenladeskript verwendet wird, werden die Werte über mehrere im group by-Zusatz bezeichnete Datensätze aggregiert.

Wird die Funktion als Diagrammformel verwendet, werden die Werte über die Diagrammdimensionen aggregiert.

Syntax:

```
TTest1w_dif (weight, value)
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
value	Die auszuwertenden Stichproben. Wird im Ladeskript für den Stichprobenwert kein Feldname angegeben, erhält das Feld automatisch die Bezeichnung **Value**.
weight	Jeder Wert in **value** geht entsprechend der zugehörigen Gewichtung einmal oder mehrfach in die Berechnung **weight** ein.

Beschränkungen:

Bei Textwerten, NULL-Werten oder fehlenden Werten im Formelwert liefert die Funktion NULL.

Beispiel:

```
TTest1w_dif( Weight, Value )
```

Siehe auch:

Erstellen eines typischen t-test-Reports (page 543)

TTest1w_lower

TTest1w_lower() liefert den aggregierten Wert für das untere Ende des Konfidenzintervalls für eine Reihe von Stichproben.

Diese Funktion bezieht sich auf t-Tests einer Stichprobe, bei denen die Werte der Stichprobe unterschiedlich gewichtet sind.

Falls die Funktion im Datenladeskript verwendet wird, werden die Werte über mehrere im group by-Zusatz bezeichnete Datensätze aggregiert.

Wird die Funktion als Diagrammformel verwendet, werden die Werte über die Diagrammdimensionen aggregiert.

Syntax:

```
TTest1w_lower (weight, value [, sig ])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
value	Die auszuwertenden Stichproben. Wird im Ladeskript für den Stichprobenwert kein Feldname angegeben, erhält das Feld automatisch die Bezeichnung **Value**.
weight	Jeder Wert in **value** geht entsprechend der zugehörigen Gewichtung einmal oder mehrfach in die Berechnung **weight** ein.
sig	Das zweiseitige Signifikanzniveau wird durch **sig** angegeben. Ist der Parameter **sig** nicht definiert, wird er auf 0,025 gesetzt, was dem gängigen Konfidenzintervall von 95 % entspricht.

Beschränkungen:

Bei Textwerten, NULL-Werten oder fehlenden Werten im Formelwert liefert die Funktion NULL.

Beispiele:

```
TTest1w_lower( Weight, Value )
TTest1w_lower( Weight, Value, 0.005 )
```

Siehe auch:

Erstellen eines typischen t-test-Reports (page 543)

TTest1w_sig

TTest1w_sig() liefert das aggregierte 2-seitige Signifikanzniveau für den Studenten-t-Test für eine Reihe von Stichproben.

Diese Funktion bezieht sich auf t-Tests einer Stichprobe, bei denen die Werte der Stichprobe unterschiedlich gewichtet sind.

Falls die Funktion im Datenladeskript verwendet wird, werden die Werte über mehrere im group by-Zusatz bezeichnete Datensätze aggregiert.

Wird die Funktion als Diagrammformel verwendet, werden die Werte über die Diagrammdimensionen aggregiert.

Syntax:

```
TTest1w_sig (weight, value)
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
value	Die auszuwertenden Stichproben. Wird im Ladeskript für den Stichprobenwert kein Feldname angegeben, erhält das Feld automatisch die Bezeichnung **Value**.
weight	Jeder Wert in **value** geht entsprechend der zugehörigen Gewichtung einmal oder mehrfach in die Berechnung **weight** ein.

Beschränkungen:

Bei Textwerten, NULL-Werten oder fehlenden Werten im Formelwert liefert die Funktion NULL.

Beispiel:

```
TTest1w_sig( Weight, Value )
```

Siehe auch:

Erstellen eines typischen t-test-Reports (page 543)

TTest1w_sterr

TTest1w_sterr() liefert den Standardfehler der Mittelwertdifferenz für den aggregierten Studenten-t-Test für eine Reihe von Stichproben.

Diese Funktion bezieht sich auf t-Tests einer Stichprobe, bei denen die Werte der Stichprobe unterschiedlich gewichtet sind.

Falls die Funktion im Datenladeskript verwendet wird, werden die Werte über mehrere im group by-Zusatz bezeichnete Datensätze aggregiert.

Wird die Funktion als Diagrammformel verwendet, werden die Werte über die Diagrammdimensionen aggregiert.

Syntax:

```
TTest1w_sterr (weight, value)
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
value	Die auszuwertenden Stichproben. Wird im Ladeskript für den Stichprobenwert kein Feldname angegeben, erhält das Feld automatisch die Bezeichnung **Value**.
weight	Jeder Wert in **value** geht entsprechend der zugehörigen Gewichtung einmal oder mehrfach in die Berechnung **weight** ein.

Beschränkungen:

Bei Textwerten, NULL-Werten oder fehlenden Werten im Formelwert liefert die Funktion NULL.

Beispiel:

```
TTest1w_sterr( Weight, Value )
```

Siehe auch:

Erstellen eines typischen t-test-Reports (page 543)

TTest1w_t

TTest1w_t() liefert den aggregierten t-Wert für eine Reihe von Stichproben.

Diese Funktion bezieht sich auf t-Tests einer Stichprobe, bei denen die Werte der Stichprobe unterschiedlich gewichtet sind.

Falls die Funktion im Datenladeskript verwendet wird, werden die Werte über mehrere im group by-Zusatz bezeichnete Datensätze aggregiert.

Wird die Funktion als Diagrammformel verwendet, werden die Werte über die Diagrammdimensionen aggregiert.

Syntax:

```
TTest1w_t ( weight, value)
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
value	Die auszuwertenden Stichproben. Wird im Ladeskript für den Stichprobenwert kein Feldname angegeben, erhält das Feld automatisch die Bezeichnung **Value**.
weight	Jeder Wert in **value** geht entsprechend der zugehörigen Gewichtung einmal oder mehrfach in die Berechnung **weight** ein.

Beschränkungen:

Bei Textwerten, NULL-Werten oder fehlenden Werten im Formelwert liefert die Funktion NULL.

Beispiel:

```
TTest1w_t( Weight, Value )
```

Siehe auch:

Erstellen eines typischen t-test-Reports (page 543)

TTest1w_upper

TTest1w_upper() liefert den aggregierten Wert für das obere Ende des Konfidenzintervalls für eine Reihe von Stichproben.

Diese Funktion bezieht sich auf t-Tests einer Stichprobe, bei denen die Werte der Stichprobe unterschiedlich gewichtet sind.

Falls die Funktion im Datenladeskript verwendet wird, werden die Werte über mehrere im group by-Zusatz bezeichnete Datensätze aggregiert.

Wird die Funktion als Diagrammformel verwendet, werden die Werte über die Diagrammdimensionen aggregiert.

Syntax:

```
TTest1w_upper (weight, value [, sig])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
value	Die auszuwertenden Stichproben. Wird im Ladeskript für den Stichprobenwert kein Feldname angegeben, erhält das Feld automatisch die Bezeichnung **Value**.
weight	Jeder Wert in **value** geht entsprechend der zugehörigen Gewichtung einmal oder mehrfach in die Berechnung **weight** ein.
sig	Das zweiseitige Signifikanzniveau wird durch **sig** angegeben. Ist der Parameter **sig** nicht definiert, wird er auf 0,025 gesetzt, was dem gängigen Konfidenzintervall von 95 % entspricht.

Beschränkungen:

Bei Textwerten, NULL-Werten oder fehlenden Werten im Formelwert liefert die Funktion NULL.

Beispiele:

```
TTest1w_upper( Weight, Value )
TTest1w_upper( Weight, Value, 0.005 )
```

Siehe auch:

Erstellen eines typischen t-test-Reports (page 543)

Z-Testfunktionen

Eine statistische Untersuchung zweier Populationsmittel. Ein z-Test von zwei Stichproben untersucht, ob die beiden Proben sich unterscheiden. Er wird in der Regel verwendet, wenn zwei normale Verteilungen über bekannte Varianzen verfügen und es sich um ein Experiment mit großem Stichprobenumfang handelt.

Die statistischen z-Testfunktionen sind nach der Art der Eingabedaten für die Funktion gruppiert.

Falls die Funktion im Datenladeskript verwendet wird, werden die Werte über mehrere im group by-Zusatz bezeichnete Datensätze aggregiert.

Wird die Funktion als Diagrammformel verwendet, werden die Werte über die Diagrammdimensionen aggregiert.

Beispiele zur Verwendung von z-test-Funktionen (page 547)

Formatfunktionen mit einer Spalte

Die folgenden Funktionen gelten für z-Tests mit einfachen Eingabedaten.

ztest_conf

ZTest_conf() liefert den aggregierten z-Wert für eine Reihe von Stichproben.

```
ZTest_conf() liefert den aggregierten z-Wert für eine Reihe von Stichproben.
(value [, sigma [, sig ])
```

ztest_dif

ZTest_dif() liefert die aggregierte Mittelwertdifferenz für den Z-Test einer Reihe von Stichproben.

```
ZTest_dif() liefert die aggregierte Mittelwertdifferenz für den Z-Test einer
Reihe von Stichproben. (value [, sigma])
```

ztest_sig

ZTest_sig() liefert das aggregierte 2-seitige Signifikanzniveau für den Z-Test für eine Reihe von Stichproben.

```
ZTest_sig() liefert das aggregierte 2-seitige Signifikanzniveau für den Z-
Test für eine Reihe von Stichproben. (value [, sigma])
```

ztest_sterr

ZTest_sterr() liefert den aggregierten Standardfehler der Mittelwertdifferenz für den Z-Test für eine Reihe von Stichproben.

```
ZTest_sterr() liefert den aggregierten Standardfehler der Mittelwertdifferenz
für den Z-Test für eine Reihe von Stichproben. (value [, sigma])
```

ztest_z

ZTest_z() liefert den aggregierten z-Wert für eine Reihe von Stichproben.

```
ZTest_z() liefert den aggregierten z-Wert für eine Reihe von Stichproben.
(value [, sigma])
```

ztest_lower

ZTest_lower() liefert den aggregierten Wert für das untere Ende des Konfidenzintervalls zweier unabhängiger Stichproben.

```
ZTest_lower() liefert den aggregierten Wert für das untere Ende des
Konfidenzintervalls zweier unabhängiger Stichproben. (grp, value [, sig [,
eq_var]])
```

ztest_upper

ZTest_upper() liefert den aggregierten Wert für das obere Ende des Konfidenzintervalls zweier unabhängiger Stichproben.

```
ZTest_upper() liefert den aggregierten Wert für das obere Ende des
Konfidenzintervalls zweier unabhängiger Stichproben. (grp, value [, sig [,
eq_var]])
```

Funktionen für gewichtetes Format mit zwei Spalten

Die folgenden fünf Funktionen beziehen sich auf z-Tests, bei denen die Werte der Stichprobe unterschiedlich gewichtet sind.

ztestw_conf

ZTestw_conf() liefert das aggregierte Z-Konfidenzintervall für eine Reihe von Stichproben.

```
ZTestw_conf() liefert das aggregierte Z-Konfidenzintervall für eine Reihe von Stichproben. (weight, value [, sigma [, sig]])
```

ztestw_dif

ZTestw_dif() liefert die aggregierte Mittelwertdifferenz für den Z-Test einer Reihe von Stichproben.

```
ZTestw_dif() liefert die aggregierte Mittelwertdifferenz für den Z-Test einer Reihe von Stichproben.  (weight, value [, sigma])
```

ztestw_lower

ZTestw_lower() liefert den aggregierten Wert für das untere Ende des Konfidenzintervalls zweier unabhängiger Stichproben.

```
ZTestw_lower() liefert den aggregierten Wert für das untere Ende des Konfidenzintervalls zweier unabhängiger Stichproben. (weight, value [, sigma])
```

ztestw_sig

ZTestw_sig() liefert das aggregierte 2-seitige Signifikanzniveau für den Z-Test für eine Reihe von Stichproben.

```
ZTestw_sig() liefert das aggregierte 2-seitige Signifikanzniveau für den Z-Test für eine Reihe von Stichproben. (weight, value [, sigma])
```

ztestw_sterr

ZTestw_sterr() liefert den aggregierten Standardfehler der Mittelwertdifferenz für den Z-Test für eine Reihe von Stichproben.

```
ZTestw_sterr() liefert den aggregierten Standardfehler der Mittelwertdifferenz für den Z-Test für eine Reihe von Stichproben. (weight, value [, sigma])
```

ztestw_upper

ZTestw_upper() liefert den aggregierten Wert für das obere Ende des Konfidenzintervalls zweier unabhängiger Stichproben.

```
ZTestw_upper() liefert den aggregierten Wert für das obere Ende des Konfidenzintervalls zweier unabhängiger Stichproben. (weight, value [, sigma])
```

ztestw_z

ZTestw_z() liefert den aggregierten z-Wert für eine Reihe von Stichproben.

```
ZTestw_z() liefert den aggregierten z-Wert für eine Reihe von Stichproben. (weight, value [, sigma])
```

ZTest_z

ZTest_z() liefert den aggregierten z-Wert für eine Reihe von Stichproben.

Falls die Funktion im Datenladeskript verwendet wird, werden die Werte über mehrere im group by-Zusatz bezeichnete Datensätze aggregiert.

Wird die Funktion als Diagrammformel verwendet, werden die Werte über die Diagrammdimensionen aggregiert.

Syntax:

```
ZTest_z(value[, sigma])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
value	Die zu interpretierenden Stichprobenwerte. In der Population wird ein Mittelwert von 0 angenommen. Soll ein anderer Mittelwert zugrunde gelegt werden, ziehen Sie diesen von den Stichprobenwerten ab.
sigma	Ist die Standardabweichung bekannt, kann sie in **sigma** angegeben werden. Fehlt **sigma**, wird die Standardabweichung der Stichprobe verwendet.

Beschränkungen:

Bei Textwerten, NULL-Werten oder fehlenden Werten im Formelwert liefert die Funktion NULL.

Beispiel:

```
ZTest_z( Value-TestValue )
```

Siehe auch:

Beispiele zur Verwendung von z-test-Funktionen (page 547)

ZTest_sig

ZTest_sig() liefert das aggregierte 2-seitige Signifikanzniveau für den Z-Test für eine Reihe von Stichproben.

Falls die Funktion im Datenladeskript verwendet wird, werden die Werte über mehrere im group by-Zusatz bezeichnete Datensätze aggregiert.

Wird die Funktion als Diagrammformel verwendet, werden die Werte über die Diagrammdimensionen aggregiert.

Syntax:

```
ZTest_sig(value[, sigma])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
value	Die zu interpretierenden Stichprobenwerte. In der Population wird ein Mittelwert von 0 angenommen. Soll ein anderer Mittelwert zugrunde gelegt werden, ziehen Sie diesen von den Stichprobenwerten ab.
sigma	Ist die Standardabweichung bekannt, kann sie in **sigma** angegeben werden. Fehlt **sigma**, wird die Standardabweichung der Stichprobe verwendet.

Beschränkungen:

Bei Textwerten, NULL-Werten oder fehlenden Werten im Formelwert liefert die Funktion NULL.

Beispiel:

```
ZTest_sig(Value-TestValue)
```

Siehe auch:

Beispiele zur Verwendung von z-test-Funktionen (page 547)

ZTest_dif

ZTest_dif() liefert die aggregierte Mittelwertdifferenz für den Z-Test einer Reihe von Stichproben.

Falls die Funktion im Datenladeskript verwendet wird, werden die Werte über mehrere im group by-Zusatz bezeichnete Datensätze aggregiert.

Wird die Funktion als Diagrammformel verwendet, werden die Werte über die Diagrammdimensionen aggregiert.

Syntax:

```
ZTest_dif(value[, sigma])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
value	Die zu interpretierenden Stichprobenwerte. In der Population wird ein Mittelwert von 0 angenommen. Soll ein anderer Mittelwert zugrunde gelegt werden, ziehen Sie diesen von den Stichprobenwerten ab.
sigma	Ist die Standardabweichung bekannt, kann sie in **sigma** angegeben werden. Fehlt **sigma**, wird die Standardabweichung der Stichprobe verwendet.

Beschränkungen:

Bei Textwerten, NULL-Werten oder fehlenden Werten im Formelwert liefert die Funktion NULL.

Beispiel:

```
ZTest_dif(Value-TestValue)
```

Siehe auch:

Beispiele zur Verwendung von z-test-Funktionen (page 547)

ZTest_sterr

ZTest_sterr() liefert den aggregierten Standardfehler der Mittelwertdifferenz für den Z-Test für eine Reihe von Stichproben.

Falls die Funktion im Datenladeskript verwendet wird, werden die Werte über mehrere im group by-Zusatz bezeichnete Datensätze aggregiert.

Wird die Funktion als Diagrammformel verwendet, werden die Werte über die Diagrammdimensionen aggregiert.

Syntax:

```
ZTest_sterr(value[, sigma])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
value	Die zu interpretierenden Stichprobenwerte. In der Population wird ein Mittelwert von 0 angenommen. Soll ein anderer Mittelwert zugrunde gelegt werden, ziehen Sie diesen von den Stichprobenwerten ab.
sigma	Ist die Standardabweichung bekannt, kann sie in **sigma** angegeben werden. Fehlt **sigma**, wird die Standardabweichung der Stichprobe verwendet.

Beschränkungen:

Bei Textwerten, NULL-Werten oder fehlenden Werten im Formelwert liefert die Funktion NULL.

Beispiel:

```
ZTest_sterr(Value-TestValue)
```

Siehe auch:

Beispiele zur Verwendung von z-test-Funktionen (page 547)

ZTest_conf

ZTest_conf() liefert den aggregierten z-Wert für eine Reihe von Stichproben.

Falls die Funktion im Datenladeskript verwendet wird, werden die Werte über mehrere im group by-Zusatz bezeichnete Datensätze aggregiert.

Wird die Funktion als Diagrammformel verwendet, werden die Werte über die Diagrammdimensionen aggregiert.

Syntax:

```
ZTest_conf(value[, sigma[, sig]])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
value	Die zu interpretierenden Stichprobenwerte. In der Population wird ein Mittelwert von 0 angenommen. Soll ein anderer Mittelwert zugrunde gelegt werden, ziehen Sie diesen von den Stichprobenwerten ab.
sigma	Ist die Standardabweichung bekannt, kann sie in **sigma** angegeben werden. Fehlt **sigma**, wird die Standardabweichung der Stichprobe verwendet.
sig	Das zweiseitige Signifikanzniveau wird durch **sig** angegeben. Ist der Parameter **sig** nicht definiert, wird er auf 0,025 gesetzt, was dem gängigen Konfidenzintervall von 95 % entspricht.

Beschränkungen:

Bei Textwerten, NULL-Werten oder fehlenden Werten im Formelwert liefert die Funktion NULL.

Beispiel:

```
ZTest_conf(Value-TestValue)
```

Siehe auch:

Beispiele zur Verwendung von z-test-Funktionen (page 547)

ZTest_lower

ZTest_lower() liefert den aggregierten Wert für das untere Ende des Konfidenzintervalls zweier unabhängiger Stichproben.

Falls die Funktion im Datenladeskript verwendet wird, werden die Werte über mehrere im group by-Zusatz bezeichnete Datensätze aggregiert.

Wird die Funktion als Diagrammformel verwendet, werden die Werte über die Diagrammdimensionen aggregiert.

Syntax:

```
ZTest_lower (grp, value [, sig [, eq_var]])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
value	Die zu interpretierenden Stichprobenwerte. Die Stichprobenwerte müssen logisch gruppiert werden, wie von exakt zwei Werten in **group** festgelegt. Wird im Ladeskript für den Stichprobenwert kein Feldname angegeben, erhält das Feld automatisch die Bezeichnung **Value**.
grp	Das Feld mit den Namen der beiden Stichprobengruppen. Wird im Ladeskript für die Gruppe kein Feldname angegeben, erhält das Feld automatisch die Bezeichnung **Type**.
sig	Das zweiseitige Signifikanzniveau wird durch **sig** angegeben. Ist der Parameter **sig** nicht definiert, wird er auf 0,025 gesetzt, was dem gängigen Konfidenzintervall von 95 % entspricht.
eq_var	Ergibt **eq_var** False (0), werden unterschiedliche Varianzen in beiden Stichproben angenommen. Ergibt **eq_var** True (1), werden gleiche Varianzen in beiden Stichproben angenommen.

Beschränkungen:

Bei Textwerten, NULL-Werten oder fehlenden Werten im Formelwert liefert die Funktion NULL.

Beispiele:

```
ZTest_lower( Group, Value )
ZTest_lower( Group, Value, sig, false )
```

Siehe auch:

Beispiele zur Verwendung von z-test-Funktionen (page 547)

ZTest_upper

ZTest_upper() liefert den aggregierten Wert für das obere Ende des Konfidenzintervalls zweier unabhängiger Stichproben.

Die Funktion bezieht sich auf t-Tests für unabhängige Stichproben.

Falls die Funktion im Datenladeskript verwendet wird, werden die Werte über mehrere im group by-Zusatz bezeichnete Datensätze aggregiert.

Wird die Funktion als Diagrammformel verwendet, werden die Werte über die Diagrammdimensionen aggregiert.

Syntax:

```
ZTest_upper (grp, value [, sig [, eq_var]])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
value	Die zu interpretierenden Stichprobenwerte. Die Stichprobenwerte müssen logisch gruppiert werden, wie von exakt zwei Werten in **group** festgelegt. Wird im Ladeskript für den Stichprobenwert kein Feldname angegeben, erhält das Feld automatisch die Bezeichnung **Value**.
grp	Das Feld mit den Namen der beiden Stichprobengruppen. Wird im Ladeskript für die Gruppe kein Feldname angegeben, erhält das Feld automatisch die Bezeichnung **Type**.
sig	Das zweiseitige Signifikanzniveau wird durch **sig** angegeben. Ist der Parameter **sig** nicht definiert, wird er auf 0,025 gesetzt, was dem gängigen Konfidenzintervall von 95 % entspricht.
eq_var	Ergibt **eq_var** False (0), werden unterschiedliche Varianzen in beiden Stichproben angenommen. Ergibt **eq_var** True (1), werden gleiche Varianzen in beiden Stichproben angenommen.

Beschränkungen:

Bei Textwerten, NULL-Werten oder fehlenden Werten im Formelwert liefert die Funktion NULL.

Beispiele:

```
ZTest_upper( Group, Value )
ZTest_upper( Group, Value, sig, false )
```

Siehe auch:

Beispiele zur Verwendung von z-test-Funktionen (page 547)

ZTestw_z

ZTestw_z() liefert den aggregierten z-Wert für eine Reihe von Stichproben.

Diese Funktion bezieht sich auf z-Tests, bei denen die Werte der Stichprobe unterschiedlich gewichtet sind.

Falls die Funktion im Datenladeskript verwendet wird, werden die Werte über mehrere im group by-Zusatz bezeichnete Datensätze aggregiert.

Wird die Funktion als Diagrammformel verwendet, werden die Werte über die Diagrammdimensionen aggregiert.

Syntax:

```
ZTestw_z (weight, value [, sigma])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
value	Die Stichprobenwerte sollten durch **value** repräsentiert sein. Es wird ein Mittelwert von 0 angenommen. Soll ein anderer Mittelwert zugrunde gelegt werden, ziehen Sie diesen von den Stichprobenwerten ab.
weight	Jeder Beispielwert in **value** geht entsprechend der zugehörigen Gewichtung einmal oder mehrfach in die Berechnung **weight** ein.
sigma	Ist die Standardabweichung bekannt, kann sie in **sigma** angegeben werden. Fehlt **sigma**, wird die Standardabweichung der Stichprobe verwendet.

Beschränkungen:

Bei Textwerten, NULL-Werten oder fehlenden Werten im Formelwert liefert die Funktion NULL.

Beispiel:

```
ZTestw_z( Weight, Value-TestValue)
```

Siehe auch:

Beispiele zur Verwendung von z-test-Funktionen (page 547)

ZTestw_sig

ZTestw_sig() liefert das aggregierte 2-seitige Signifikanzniveau für den Z-Test für eine Reihe von Stichproben.

Diese Funktion bezieht sich auf z-Tests, bei denen die Werte der Stichprobe unterschiedlich gewichtet sind.

Falls die Funktion im Datenladeskript verwendet wird, werden die Werte über mehrere im group by-Zusatz bezeichnete Datensätze aggregiert.

Wird die Funktion als Diagrammformel verwendet, werden die Werte über die Diagrammdimensionen aggregiert.

Syntax:

```
ZTestw_sig (weight, value [, sigma])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
value	Die Stichprobenwerte sollten durch **value** repräsentiert sein. Es wird ein Mittelwert von 0 angenommen. Soll ein anderer Mittelwert zugrunde gelegt werden, ziehen Sie diesen von den Stichprobenwerten ab.
weight	Jeder Beispielwert in **value** geht entsprechend der zugehörigen Gewichtung einmal oder mehrfach in die Berechnung **weight** ein.
sigma	Ist die Standardabweichung bekannt, kann sie in **sigma** angegeben werden. Fehlt **sigma**, wird die Standardabweichung der Stichprobe verwendet.

Beschränkungen:

Bei Textwerten, NULL-Werten oder fehlenden Werten im Formelwert liefert die Funktion NULL.

Beispiel:

```
ZTestw_sig( Weight, Value-TestValue)
```

Siehe auch:

Beispiele zur Verwendung von z-test-Funktionen (page 547)

ZTestw_dif

ZTestw_dif() liefert die aggregierte Mittelwertdifferenz für den Z-Test einer Reihe von Stichproben.

Diese Funktion bezieht sich auf z-Tests, bei denen die Werte der Stichprobe unterschiedlich gewichtet sind.

Falls die Funktion im Datenladeskript verwendet wird, werden die Werte über mehrere im group by-Zusatz bezeichnete Datensätze aggregiert.

Wird die Funktion als Diagrammformel verwendet, werden die Werte über die Diagrammdimensionen aggregiert.

Syntax:

```
ZTestw_dif ( weight, value [, sigma])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
value	Die Stichprobenwerte sollten durch **value** repräsentiert sein. Es wird ein Mittelwert von 0 angenommen. Soll ein anderer Mittelwert zugrunde gelegt werden, ziehen Sie diesen von den Stichprobenwerten ab.
weight	Jeder Beispielwert in **value** geht entsprechend der zugehörigen Gewichtung einmal oder mehrfach in die Berechnung **weight** ein.
sigma	Ist die Standardabweichung bekannt, kann sie in **sigma** angegeben werden. Fehlt **sigma**, wird die Standardabweichung der Stichprobe verwendet.

Beschränkungen:

Bei Textwerten, NULL-Werten oder fehlenden Werten im Formelwert liefert die Funktion NULL.

Beispiel:

```
ZTestw_dif( Weight, Value-TestValue)
```

Siehe auch:

Beispiele zur Verwendung von z-test-Funktionen (page 547)

ZTestw_sterr

ZTestw_sterr() liefert den aggregierten Standardfehler der Mittelwertdifferenz für den Z-Test für eine Reihe von Stichproben.

Diese Funktion bezieht sich auf z-Tests, bei denen die Werte der Stichprobe unterschiedlich gewichtet sind.

Falls die Funktion im Datenladeskript verwendet wird, werden die Werte über mehrere im group by-Zusatz bezeichnete Datensätze aggregiert.

Wird die Funktion als Diagrammformel verwendet, werden die Werte über die Diagrammdimensionen aggregiert.

Syntax:

```
ZTestw_sterr (weight, value [, sigma])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
value	Die Stichprobenwerte sollten durch **value** repräsentiert sein. Es wird ein Mittelwert von 0 angenommen. Soll ein anderer Mittelwert zugrunde gelegt werden, ziehen Sie diesen von den Stichprobenwerten ab.
weight	Jeder Beispielwert in **value** geht entsprechend der zugehörigen Gewichtung einmal oder mehrfach in die Berechnung **weight** ein.
sigma	Ist die Standardabweichung bekannt, kann sie in **sigma** angegeben werden. Fehlt **sigma**, wird die Standardabweichung der Stichprobe verwendet.

Beschränkungen:

Bei Textwerten, NULL-Werten oder fehlenden Werten im Formelwert liefert die Funktion NULL.

Beispiel:

```
ZTestw_sterr( Weight, Value-TestValue)
```

Siehe auch:

Beispiele zur Verwendung von z-test-Funktionen (page 547)

ZTestw_conf

ZTestw_conf() liefert das aggregierte Z-Konfidenzintervall für eine Reihe von Stichproben.

Diese Funktion bezieht sich auf z-Tests, bei denen die Werte der Stichprobe unterschiedlich gewichtet sind.

Falls die Funktion im Datenladeskript verwendet wird, werden die Werte über mehrere im group by-Zusatz bezeichnete Datensätze aggregiert.

Wird die Funktion als Diagrammformel verwendet, werden die Werte über die Diagrammdimensionen aggregiert.

Syntax:

```
ZTest_conf(weight, value[, sigma[, sig]])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
value	Die zu interpretierenden Stichprobenwerte. In der Population wird ein Mittelwert von 0 angenommen. Soll ein anderer Mittelwert zugrunde gelegt werden, ziehen Sie diesen von den Stichprobenwerten ab.
weight	Jeder Beispielwert in **value** geht entsprechend der zugehörigen Gewichtung einmal oder mehrfach in die Berechnung **weight** ein.
sigma	Ist die Standardabweichung bekannt, kann sie in **sigma** angegeben werden. Fehlt **sigma**, wird die Standardabweichung der Stichprobe verwendet.
sig	Das zweiseitige Signifikanzniveau wird durch **sig** angegeben. Ist der Parameter **sig** nicht definiert, wird er auf 0,025 gesetzt, was dem gängigen Konfidenzintervall von 95 % entspricht.

Beschränkungen:

Bei Textwerten, NULL-Werten oder fehlenden Werten im Formelwert liefert die Funktion NULL.

Beispiel:

```
ZTestw_conf( Weight, Value-TestValue)
```

Siehe auch:

Beispiele zur Verwendung von z-test-Funktionen (page 547)

ZTestw_lower

ZTestw_lower() liefert den aggregierten Wert für das untere Ende des Konfidenzintervalls zweier unabhängiger Stichproben.

Falls die Funktion im Datenladeskript verwendet wird, werden die Werte über mehrere im group by-Zusatz bezeichnete Datensätze aggregiert.

Wird die Funktion als Diagrammformel verwendet, werden die Werte über die Diagrammdimensionen aggregiert.

Syntax:

```
ZTestw_lower (grp, value [, sig [, eq_var]])
```

Rückgabe Datentyp: numerisch

Argumente:

Argument	Beschreibung
value	Die zu interpretierenden Stichprobenwerte. Die Stichprobenwerte müssen logisch gruppiert werden, wie von exakt zwei Werten in **group** festgelegt. Wird im Ladeskript für den Stichprobenwert kein Feldname angegeben, erhält das Feld automatisch die Bezeichnung **Value**.
grp	Das Feld mit den Namen der beiden Stichprobengruppen. Wird im Ladeskript für die Gruppe kein Feldname angegeben, erhält das Feld automatisch die Bezeichnung **Type**.
sig	Das zweiseitige Signifikanzniveau wird durch **sig** angegeben. Ist der Parameter **sig** nicht definiert, wird er auf 0,025 gesetzt, was dem gängigen Konfidenzintervall von 95 % entspricht.
eq_var	Ergibt **eq_var** False (0), werden unterschiedliche Varianzen in beiden Stichproben angenommen. Ergibt **eq_var** True (1), werden gleiche Varianzen in beiden Stichproben angenommen.

Beschränkungen:

Bei Textwerten, NULL-Werten oder fehlenden Werten im Formelwert liefert die Funktion NULL.

Beispiele:

```
ZTestw_lower( Group, Value )
ZTestw_lower( Group, Value, sig, false )
```

Siehe auch:

Beispiele zur Verwendung von z-test-Funktionen (page 547)

ZTestw_upper

ZTestw_upper() liefert den aggregierten Wert für das obere Ende des Konfidenzintervalls zweier unabhängiger Stichproben.

Die Funktion bezieht sich auf t-Tests für unabhängige Stichproben.

Falls die Funktion im Datenladeskript verwendet wird, werden die Werte über mehrere im group by-Zusatz bezeichnete Datensätze aggregiert.

Wird die Funktion als Diagrammformel verwendet, werden die Werte über die Diagrammdimensionen aggregiert.

Syntax:

```
ZTestw_upper (grp, value [, sig [, eq_var]])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
value	Die zu interpretierenden Stichprobenwerte. Die Stichprobenwerte müssen logisch gruppiert werden, wie von exakt zwei Werten in **group** festgelegt. Wird im Ladeskript für den Stichprobenwert kein Feldname angegeben, erhält das Feld automatisch die Bezeichnung **Value**.
grp	Das Feld mit den Namen der beiden Stichprobengruppen. Wird im Ladeskript für die Gruppe kein Feldname angegeben, erhält das Feld automatisch die Bezeichnung **Type**.
sig	Das zweiseitige Signifikanzniveau wird durch **sig** angegeben. Ist der Parameter **sig** nicht definiert, wird er auf 0,025 gesetzt, was dem gängigen Konfidenzintervall von 95 % entspricht.
eq_var	Ergibt **eq_var** False (0), werden unterschiedliche Varianzen in beiden Stichproben angenommen. Ergibt **eq_var** True (1), werden gleiche Varianzen in beiden Stichproben angenommen.

Beschränkungen:

Bei Textwerten, NULL-Werten oder fehlenden Werten im Formelwert liefert die Funktion NULL.

Beispiele:

```
ZTestw_upper( Group, Value )
ZTestw_upper( Group, Value, sig, false )
```

Siehe auch:

Beispiele zur Verwendung von z-test-Funktionen (page 547)

Aggregierungsfunktionen für statistische Tests – Beispiele

Dieser Abschnitt enthält Beispiele statistischer Testfunktionen für Diagramme und das Datenladeskript.

Beispiele zur Verwendung der chi2-test-Funktionen in Diagrammen

Die Funktionen chi2-test werden zum Ermitteln von Werten verwendet, die mit der chi2-Statistikanalyse verbunden sind.

In diesem Abschnitt wird das Erstellen von Visualisierungen mithilfe von Beispieldaten beschrieben. Dies ermöglicht das Ermitteln der Werte von chi2-Verteilungstestfunktionen in Qlik Sense. Beschreibungen für Syntax und Argumente sind in den einzelnen Themen zu den Diagrammfunktionen chi2-test zu finden.

Laden der Daten für die Stichproben

Es gibt drei verschiedene Gruppen mit Stichprobendaten, die drei verschiedene Statistikstichproben für das Laden in das Skript beschreiben.

Gehen Sie folgendermaßen vor:

1. Erstellen Sie eine neue App.

   Geben Sie im Datenimport Folgendes ein:
   ```
   // Sample_1 data is pre-aggregated... Note: make sure you set your DecimalSep='.' at the top of the script.
   ```
2. ```
   Sample_1:

   LOAD * inline [

   Grp,Grade,Count

   I,A,15

   I,B,7

   I,C,9

   I,D,20

   I,E,26

   I,F,19

   II,A,10

   II,B,11

   II,C,7

   II,D,15

   II,E,21

   II,F,16

   ];

   // Sample_2 data is pre-aggregated: If raw data is used, it must be aggregated using count()...

   Sample_2:

   LOAD * inline [

   Sex,Opinion,OpCount
   ```
   ```

```
1,2,58

1,1,11

1,0,10

2,2,35

2,1,25

2,0,23 ] (delimiter is ',');

// Sample_3a data is transformed using the crosstable statement...

Sample_3a:

crosstable(Gender, Actual) LOAD

Description,

[Men (Actual)] as Men,

[Women (Actual)] as Women;

LOAD * inline [

Men (Actual),Women (Actual),Description

58,35,Agree

11,25,Neutral

10,23,Disagree ] (delimiter is ',');

// Sample_3b data is transformed using the crosstable statement...

Sample_3b:

crosstable(Gender, Expected) LOAD

Description,

[Men (Expected)] as Men,

[Women (Expected)] as Women;

LOAD * inline [

Men (Expected),Women (Expected),Description

45.35,47.65,Agree

17.56,18.44,Neutral
```

```
16.09,16.91,Disagree ] (delimiter is ',');

// Sample_3a and Sample_3b will result in a (fairly harmless) Synthetic Key...
```

3. Klicken Sie auf , um Daten zu laden.

Erstellen der Visualisierungen für die Diagrammfunktion chi2-test

Beispiel: Stichprobe 1

Gehen Sie folgendermaßen vor:

1. Klicken Sie im Dateneditor auf , um zur App-Ansicht zu wechseln, und klicken Sie dann auf das Arbeitsblatt, das Sie vorher erstellt haben.
   Die Arbeitsblatt-Ansicht wird geöffnet.
2. Klicken Sie auf **Arbeitsblatt bearbeiten**, um das Arbeitsblatt zu bearbeiten.
3. Fügen Sie über **Diagramme** eine Tabelle und über **Felder** die Dimensionen Grp, Grade und Count hinzu.
   In dieser Tabelle werden die Stichprobendaten angezeigt.
4. Fügen Sie eine weitere Tabelle mit folgender Formel als Dimension hinzu.
   `ValueList('p','df','Chi2')`
   Dadurch wird die Funktion für synthetische Dimensionen verwendet, um Beschriftungen für die Dimensionen mit den Namen der drei chi2-test-Funktionen zu erstellen.
5. Fügen Sie die folgende Formel als Kennzahl zur Tabelle hinzu.

```
IF(ValueList('p','df','Chi2')='p',Chi2Test_p(Grp,Grade,Count),

IF(ValueList('p','df','Chi2')='df',Chi2Test_df(Grp,Grade,Count),

Chi2Test_Chi2(Grp,Grade,Count)))
```

   Dadurch wird der Ergebniswert jeder chi2-test-Funktion in der Tabelle neben der jeweils assoziierten synthetischen Dimension angezeigt.
6. Legen Sie das **Zahlenformat** der Kennzahl als **Zahl** und zudem **3Wichtige Zahlen** fest.

> *In der Formel für die Kennzahl können Sie stattdessen die folgende Formel verwenden:* `Pick(Match(ValueList('p','df','Chi2'),'p','df','Chi2'),Chi2Test_p(Grp,Grade,Count),Chi2Test_df(Grp,Grade,Count),Chi2Test_Chi2(Grp,Grade,Count))`

Ergebnis:

Die Tabelle, die sich für die chi2-test-Funktionen für die Daten von Stichprobe 1 ergibt, enthält die folgenden Werte:

Ergebnistabelle

p	df	Chi2
0.820	5	2.21

Beispiel: Stichprobe 2

Gehen Sie folgendermaßen vor:

1. Fügen Sie in das Arbeitsblatt, das Sie im Beispiel von Stichprobe 1 bearbeitet haben, über **Diagramme** eine Tabelle und über **Felder** die Dimensionen Sex, Opinion und OpCount hinzu.
2. Erstellen Sie eine Kopie der Ergebnistabelle von Stichprobe 1 mithilfe der Befehle **Kopieren** und **Einfügen**. Bearbeiten Sie die Formel in der Kennzahl und ersetzen Sie die Argumente in allen drei chi2-test-Funktionen durch die Namen der Felder, die für die Daten von Stichprobe 2 verwendet werden, zum Beispiel: `Chi2Test_p(Sex,Opinion,OpCount)`.

Ergebnis:

Die Tabelle, die sich für die chi2-test-Funktionen für die Daten von Stichprobe 2 ergibt, enthält die folgenden Werte:

Ergebnistabelle

p	df	Chi2
0.000309	2	16.2

Beispiel: Stichprobe 3

Gehen Sie folgendermaßen vor:

1. Erstellen Sie genauso wie in den Beispielen für die Daten von Stichprobe 1 und Stichprobe 2 zwei weitere Tabellen. Verwenden Sie in der Dimensionstabelle die folgenden Felder als Dimensionen: Gender, Description, Actual und Expected.
2. In der Ergebnistabelle verwenden Sie die Namen und Felder der Daten von Stichprobe 3, zum Beispiel: `Chi2Test_p(Gender,Description,Actual,Expected)`.

Ergebnis:

Die Tabelle, die sich für die chi2-test-Funktionen für die Daten von Stichprobe 3 ergibt, enthält die folgenden Werte:

Ergebnistabelle

p	df	Chi2
0.000308	2	16.2

Beispiele zur Verwendung der chi2-test-Funktionen im Datenladeskript

Die Funktionen chi2-test werden zum Ermitteln von Werten verwendet, die mit der chi2-Statistikanalyse verbunden sind. In diesem Abschnitt wird beschrieben, wie die in Qlik Sense verfügbaren chi2-Verteilungstestfunktionen im Datenladeskript verwendet werden. Beschreibungen für Syntax und Argumente sind in den einzelnen Themen zu den Skriptfunktionen chi2-test zu finden.

Diese Beispiele nutzen eine Tabelle mit Schülern, die in zwei Schülergruppen (I und II) mit den Noten (A-F) beurteilt wurden.

Data table

Group	A	B	C	D	E	F
I	15	7	9	20	26	19
II	10	11	7	15	21	16

Laden der Beispieldaten

Gehen Sie folgendermaßen vor:

1. Erstellen Sie eine neue App.

   Geben Sie im Dateneditor Folgendes ein:
   ```
   // Sample_1 data is pre-aggregated... Note: make sure you set your DecimalSep='.' at the top of the script.
   ```
2. ```
   Sample_1:

   LOAD * inline [

   Grp,Grade,Count

   I,A,15

   I,B,7

   I,C,9

   I,D,20

   I,E,26

   I,F,19

   II,A,10

   II,B,11

   II,C,7

   II,D,15

   II,E,21

   II,F,16

   ];
   ```
3. Klicken Sie auf , um Daten zu laden.

Sie haben die Beispieldaten jetzt geladen.
   ```

Laden der chi2-test-Funktionswerte

Jetzt laden wir die chi2-test-Werte basierend auf den Beispieldaten in eine neue Tabelle, gruppiert nach Grp.

Gehen Sie folgendermaßen vor:

1. Fügen Sie im Datenladeeditor die folgende Informationen zum Ende des Skripts hinzu:

```
// Sample_1 data is pre-aggregated... Note: make sure you set your DecimalSep='.' at the top of the script.

Chi2_table:

LOAD Grp,

Chi2Test_chi2(Grp, Grade, Count) as chi2,

Chi2Test_df(Grp, Grade, Count) as df,

Chi2Test_p(Grp, Grade, Count) as p

resident Sample_1 group by Grp;
```

2. Klicken Sie auf , um Daten zu laden.

Sie haben die chi2-test-Werte jetzt in eine Tabelle mit dem Namen Chi2_table geladen.

Ergebnisse

Sie können die sich daraus ergebenden chi2-test-Werte im Datenmodell unter **Vorschau** ansehen und sie sollten folgendermaßen aussehen:

Results

Grp	chi2	df	p
I	16.00	5	0.007
II	9.40	5	0.094

Erstellen eines typischen t-test-Reports

Ein typischer t-test-Report kann Tabellen mit **Group Statistics**- und **Independent Samples Test**-Ergebnissen enthalten.

In den folgenden Abschnitten erstellen wir diese Tabellen mit Qlik Senset-test-Funktionen, die auf zwei unabhängige Mustergruppen, Observation und Comparison, angewendet werden. Die entsprechenden Tabellen für diese Beispiele sehen wie folgt aus:

Gruppenstatistiken

Type	N	Mean	Standard Deviation	Standard Error Mean
Comparison	20	11.95	14.61245	3.2674431
Observation	20	27.15	12.507997	2.7968933

Unabhängiger Beispieltest

Type	conf	t	df	Sig. (2-tailed)	Mean Difference	Standard Error Difference	95% Confidence Interval (Lower)	95% Confidence Interval (Upper)
Equal Variance not Assumed	0	3.534	37.116717335823	0.001	15.2	4.30101	6.48625	23.9137
Equal Variance Assumed	8.706939	3.534	38	0.001	15.2	4.30101	6.49306	23.9069

Laden der Beispieldaten

Gehen Sie folgendermaßen vor:

1. Erstellen Sie eine neue App mit einem neuen Arbeitsblatt.
2. Geben Sie Folgendes in den Dateneditor ein:
```
Table1:
Crosstable (Type, Value)
Load recno() as ID, * inline [
Observation|Comparison
35|2
40|27
12|38
15|31
21|1
14|19
46|1
10|34
28|3
48|1
16|2
30|3
32|2
48|1
31|2
22|1
12|3
39|29
19|37
25|2 ] (delimiter is '|');
```
   In diesem Ladeskript ist **recno()** enthalten, da **crosstable** drei Argumente erfordert. **recno()** liefert also nur ein zusätzliches Argument, in diesem Fall eine ID für jede Zeile. Ohne dieses Argument werden keine **Comparison**-Stichprobenwerte geladen.
3. Klicken Sie auf , um Daten zu laden.

Erstellen der Tabelle Group statistics

Gehen Sie folgendermaßen vor:

1. Klicken Sie im Dateneditor auf , um zur App-Ansicht zu wechseln, und klicken Sie dann auf das Arbeitsblatt, das Sie vorher erstellt haben.
   Die Arbeitsblatt-Ansicht wird geöffnet.
2. Klicken Sie auf **Arbeitsblatt bearbeiten**, um das Arbeitsblatt zu bearbeiten.
3. Fügen Sie über **Diagramme** eine Tabelle und über **Felder** Type als Dimension zur Tabelle hinzu.
4. Fügen Sie die folgenden Formeln als Kennzahlen hinzu:

   Beispielformeln

Bezeichnung	**Formel**
N	Count(Value)
Mean	Avg(Value)
Standard Deviation	Stdev(Value)
Standard Error Mean	Sterr(Value)

5. Klicken Sie auf **Sortieren** und vergewissern Sie sich, dass Type am Anfang der Sortierliste steht.

Ergebnis:

Eine Group statistics-Tabelle für diese Beispiele sieht wie folgt aus:

Gruppenstatistiken

Type	**N**	**Mean**	**Standard Deviation**	**Standard Error Mean**
Comparison	20	11.95	14.61245	3.2674431
Observation	20	27.15	12.507997	2.7968933

Erstellen der Tabelle Independent sample test

Gehen Sie folgendermaßen vor:

1. Klicken Sie auf **Arbeitsblatt bearbeiten**, um das Arbeitsblatt zu bearbeiten.
2. Fügen Sie über **Diagramme** eine Tabelle mit der folgenden Formel als Dimension zur Tabelle `=ValueList (Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1))` hinzu und geben Sie ihr die Bezeichnung „Typ“.

3. Fügen Sie die folgenden Formeln als Kennzahlen hinzu:

Beispielformeln

Bezeichnung	Formel
conf	if(ValueList (Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1)),TTest_conf(Type, Value),TTest_conf(Type, Value, 0))
t	if(ValueList (Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1)),TTest_t(Type, Value),TTest_t(Type, Value, 0))
df	if(ValueList (Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1)),TTest_df(Type, Value),TTest_df(Type, Value, 0))
Sig. (2-tailed)	if(ValueList (Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1)),TTest_sig(Type, Value),TTest_sig(Type, Value, 0))
Mean Difference	TTest_dif(Type, Value)
Standard Error Difference	if(ValueList (Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1)),TTest_sterr(Type, Value),TTest_sterr(Type, Value, 0))
95% Confidence Interval (Lower)	if(ValueList (Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1)),TTest_lower(Type, Value,(1-(95)/100)/2),TTest_lower(Type, Value, (1-(95)/100)/2, 0))
95% Confidence Interval (Upper)	if(ValueList (Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1)),TTest_upper(Type, Value,(1-(95)/100)/2),TTest_upper(Type, Value, (1-(95)/100)/2, 0))

Ergebnis:

Unabhängiger Beispieltest

Type	conf	t	df	Sig. (2-tailed)	Mean Difference	Standard Error Difference	95% Confidence Interval (Lower)	95% Confidence Interval (Upper)
Equal Variance not Assumed	0	3.534	37.116717335823	0.001	15.2	4.30101	6.48625	23.9137
Equal Variance Assumed	8.706939	3.534	38	0.001	15.2	4.30101	6.49306	23.9069

Beispiele zur Verwendung von z-test-Funktionen

Die Funktionen z-test werden bei bekannter Varianz verwendet, um mit der z-test-Statistikanalyse assoziierte Werte für große Datenstichproben, normalerweise größer als 30, zu ermitteln.

In diesem Abschnitt wird das Erstellen von Visualisierungen mithilfe von Beispieldaten beschrieben. Dies ermöglicht das Ermitteln der Werte von z-test-Funktionen in Qlik Sense. Beschreibungen für Syntax und Argumente sind in den einzelnen Themen zu den Diagrammfunktionen z-test zu finden.

Laden der Beispieldaten

Die hier verwendeten Stichprobendaten sind dieselben wie in den Beispielen für die t-test-Funktionen. Der Umfang der Beispieldaten wäre normalerweise zu klein für die Z-Test-Analyse, genügt aber zur Erläuterung der verschiedenen z-test-Funktionen in Qlik Sense.

Gehen Sie folgendermaßen vor:

1. Erstellen Sie eine neue App mit einem neuen Arbeitsblatt.

   Wenn Sie bereits eine App für die t-test-Funktionen erstellt haben, können Sie diese verwenden und ein neues Arbeitsblatt für diese Funktionen erstellen.

2. Geben Sie im Dateneditor Folgendes ein:

```
Table1:
Crosstable (Type, Value)
Load recno() as ID, * inline [
Observation|Comparison
35|2
40|27
12|38
15|31
21|1
14|19
46|1
10|34
28|3
48|1
16|2
30|3
32|2
48|1
31|2
22|1
12|3
39|29
19|37
25|2 ] (delimiter is '|');
```

   In diesem Ladeskript ist **recno()** enthalten, da **crosstable** drei Argumente erfordert. **recno()** liefert also nur ein zusätzliches Argument, in diesem Fall eine ID für jede Zeile. Ohne dieses Argument werden

keine **Comparison**-Stichprobenwerte geladen.

3. Klicken Sie auf , um Daten zu laden.

Erstellen der Tabelle z-test

Gehen Sie folgendermaßen vor:

1. Klicken Sie im Dateneditor auf , um zur App-Ansicht zu wechseln, und klicken Sie dann auf das Arbeitsblatt, das Sie oben erstellt haben.
   Die Arbeitsblatt-Ansicht wird geöffnet.
2. Klicken Sie auf **Arbeitsblatt bearbeiten**, um das Arbeitsblatt zu bearbeiten.
3. Fügen Sie über **Diagramme** eine Tabelle und über **Felder** die Dimension Type hinzu.
4. Fügen Sie die folgenden Formeln als Kennzahlen zur Tabelle hinzu

Beispielformeln

Bezeichnung	Formel
ZTest Conf	ZTest_conf(Value)
ZTest Dif	ZTest_dif(Value)
ZTest Sig	ZTest_sig(Value)
ZTest Sterr	ZTest_sterr(Value)
ZTest Z	ZTest_z(Value)

*Stellen Sie gegebenenfalls das Zahlenformat der Kennzahlen so ein, dass nur die relevanten Werte angezeigt werden. Die Tabelle lässt sich einfacher lesen, wenn Sie für das Zahlenformat für die meisten Kennzahlen **Nummer>Einfach** anstelle von **Auto** auswählen. Verwenden Sie aber für ZTest Sig beispielsweise das Zahlenformat: **Benutzerdefiniert**, und passen Sie dann das Zahlenformat auf **#.######** an.*

Ergebnis:

Die Tabelle, die sich für die z-test-Funktionen für die Daten der Stichprobe ergibt, enthält die folgenden Werte:

z-test Ergebnistabelle

Type	ZTest Conf	ZTest Dif	ZTest Sig	ZTest Sterr	ZTest Z
Comparison	6.40	11.95	0.000123	3.27	3.66
Observation	5.48	27.15	0.000000	2.80	9.71

Erstellen der Tabelle z-testw

Die z-testw-Funktionen werden dann verwendet, wenn die Werte der Stichprobe in zwei Spalten angegeben und unterschiedlich gewichtet sind. Die Formeln erfordern einen Wert für das Argument weight.

In den hier angegebenen Beispielen wird durchwegs 2 verwendet, aber Sie könnten auch eine Formel einsetzen, wodurch ein Wert für weight für jede Beobachtung erstellt würde.

Gehen Sie folgendermaßen vor:

1. Klicken Sie im Dateneditor auf , um zur App-Ansicht zu wechseln, und klicken Sie dann auf das Arbeitsblatt, das Sie oben erstellt haben.
   Die Arbeitsblatt-Ansicht wird geöffnet.
2. Klicken Sie auf **Arbeitsblatt bearbeiten**, um das Arbeitsblatt zu bearbeiten.
3. Fügen Sie über **Diagramme** eine Tabelle und über **Felder** die Dimension Type hinzu.
4. Fügen Sie die folgenden Formeln als Kennzahlen zur Tabelle hinzu:

   Beispielformeln

Bezeichnung	Formel
ZTestw Conf	ZTestw_conf(2,Value)
ZTestw Dif	ZTestw_dif(2,Value)
ZTestw Sig	ZTestw_sig(2,Value)
ZTestw Sterr	ZTestw_sterr(2,Value)
ZTestw Z	ZTestw_z(2,Value)

   Verwenden Sie das gleiche Zahlenformat wie im z-test-Funktionsbeispiel.

Ergebnis:

Die Tabelle, die sich für die z-testw-Funktionen ergibt, enthält die folgenden Werte:

z-testw Ergebnistabelle

Type	ZTestw Conf	ZTestw Dif	ZTestw Sig	ZTestw Sterr	ZTestw Z
Comparison	4.47	11.95	8.037185e-08	2.28	5.24
Observation	3.83	27.15	0	1.95	13.91

String-Aggregierungsfunktionen

In diesem Abschnitt werden Aggregierungsfunktionen mit String-Bezug beschrieben.

Jede Funktion wird nach der Übersicht genauer beschrieben. Sie können auch auf den Funktionsnamen in der Syntax klicken, um direkt auf die Details zu der spezifischen Funktion zuzugreifen.

String-Aggregierungsfunktionen im Datenladeskript verwenden

Concat

Concat() wird zur Kombination von Stringwerten verwendet. Die Skriptfunktion liefert die Stringverkettung aller Werte über die im **group by**-Zusatz bezeichneten Datensätze.

```
Concat ([ distinct ] expression [, delimiter [, sort-weight]])
```

FirstValue

FirstValue() liefert den Wert, der zuerst aus den durch die Formel definierten Datensätzen geladen wurde, sortiert nach der Bedingung **group by**.

Diese Funktion steht nur als Skriptfunktion zur Verfügung.

```
FirstValue (expression)
```

LastValue

LastValue() liefert den Wert, der zuletzt aus den durch die Formel definierten Datensätzen geladen wurde, sortiert nach der Bedingung **group by**.

Diese Funktion steht nur als Skriptfunktion zur Verfügung.

```
LastValue (expression)
```

MaxString

MaxString() sucht nach Stringwerten in der Formel und gibt den letzten Textwert in alphabetischer Reihenfolge über mehrere Datensätze hinweg zurück, wie in einer **group by**-Bedingung definiert.

```
MaxString (expression )
```

MinString

MinString() sucht nach Stringwerten in der Formel und gibt den ersten Textwert in alphabetischer Reihenfolge über mehrere Datensätze hinweg zurück, wie in einer **group by**-Bedingung definiert.

```
MinString (expression )
```

String-Aggregierung in Diagrammen

Die folgenden Diagrammfunktionen dienen zur String-Aggregierung in Diagrammen.

Concat

Concat() wird zur Kombination von Stringwerten verwendet. Die Funktion liefert die Stringverkettung aller Werte der über alle Dimensionen interpretierten Formel.

```
Concat - Diagrammfunktion({[SetExpression] [DISTINCT] [TOTAL [<fld{, fld}>]]
string[, delimiter[, sort_weight]])
```

MaxString

MaxString() sucht nach Stringwerten in der Formel oder im Feld und liefert den letzten Textwert in alphabetischer Reihenfolge.

```
MaxString - Diagrammfunktion({[SetExpression] [TOTAL [<fld{, fld}>]]} expr)
```

MinString

MinString() sucht nach Stringwerten in der Formel oder im Feld und liefert den ersten Textwert in alphabetischer Reihenfolge.

```
MinString - Diagrammfunktion({[SetExpression] [TOTAL [<fld {, fld}>]]} expr)
```

Concat

Concat() wird zur Kombination von Stringwerten verwendet. Die Skriptfunktion liefert die Stringverkettung aller Werte über die im **group by**-Zusatz bezeichneten Datensätze.

Syntax:

```
Concat ([ distinct ] string [, delimiter [, sort-weight]])
```

Rückgabe Datentyp: String

Argumente:

Die Formel oder das Feld mit dem String, der verarbeitet werden soll.

Argumente

Argument	Beschreibung
string	Die Formel oder das Feld mit dem String, der verarbeitet werden soll.
delimiter	Jeder Wert kann durch den String im delimiter abgetrennt werden.
sort-weight	Die Sortierung der Werte wird, sofern vorhanden, durch den Wert der Dimension **sort-weight** definiert. Der String, der dem niedrigsten Wert entspricht, wird in der Sortierung zuerst angezeigt..
distinct	Ist der Formel das Wort **distinct** vorangestellt, werden Dubletten nicht berücksichtigt.

Beispiele und Ergebnisse:

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Fügen Sie einem Arbeitsblatt in Ihrer App dann die Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um das Ergebnis anzuzeigen.

Beispiele und Ergebnisse

Beispiel	Ergebnis	Ergebnisse nach Hinzufügen zu einem Arbeitsblatt
TeamData: LOAD * inline [SalesGroup|Team|Date|Amount East|Gamma|01/05/2013|20000 East|Gamma|02/05/2013|20000 West|Zeta|01/06/2013|19000 East|Alpha|01/07/2013|25000 East|Delta|01/08/2013|14000 West|Epsilon|01/09/2013|17000 West|Eta|01/10/2013|14000 East|Beta|01/11/2013|20000 West|Theta|01/12/2013|23000] (delimiter is '|'); Concat1: LOAD SalesGroup,Concat(Team) as TeamConcat1 Resident TeamData Group By SalesGroup;	SalesGroup East West	TeamConcat1 AlphaBetaDeltaGammaGamma EpsilonEtaThetaZeta
Vorgabe: Die Tabelle **TeamData** wird wie im vorherigen Beispiel geladen: LOAD SalesGroup,Concat(distinct Team,'-') as TeamConcat2 Resident TeamData Group By SalesGroup;	SalesGroup East West	TeamConcat2 Alpha-Beta-Delta-Gamma Epsilon-Eta-Theta-Zeta
Vorgabe: Die Tabelle **TeamData** wird wie im vorherigen Beispiel geladen. Weil das Argument für **sort-weight** hinzugefügt wird, werden die Ergebnisse nach dem Wert der Dimension Amount sortiert: LOAD SalesGroup,Concat(distinct Team,'-',Amount) as TeamConcat2 Resident TeamData Group By SalesGroup;	SalesGroup East West	TeamConcat2 Delta-Beta-Gamma-Alpha Eta-Epsilon-Zeta-Theta

Concat - Diagrammfunktion

Concat() wird zur Kombination von Stringwerten verwendet. Die Funktion liefert die Stringverkettung aller Werte der über alle Dimensionen interpretierten Formel.

Syntax:

```
Concat({[SetExpression] [DISTINCT] [TOTAL [<fld{, fld}>]]} string[, delimiter
[, sort_weight]])
```

Rückgabe Datentyp: String

Argumente:

Argumente

Argument	Beschreibung
string	Die Formel oder das Feld mit dem String, der verarbeitet werden soll.
delimiter	Jeder Wert kann durch den String im delimiter abgetrennt werden.
sort-weight	Die Sortierung der Werte wird, sofern vorhanden, durch den Wert der Dimension **sort-weight** definiert. Der String, der dem niedrigsten Wert entspricht, wird in der Sortierung zuerst angezeigt..
SetExpression	Standardmäßig berechnet sich die Aggregierungsfunktion über alle wählbaren Werte. Alternativ können Sie die der Berechnung zugrunde liegenden Werte über die Auswahlformel bestimmen.
DISTINCT	Der Zusatz **DISTINCT** vor den Funktionsargumenten bewirkt, dass bei der Auswertung der Funktionsargumente entstehende Duplikate nicht berücksichtigt werden.
TOTAL	Der Zusatz **TOTAL** vor der Funktion bewirkt, dass die Berechnung über alle ausgewählten bzw. wählbaren Werte erfolgt, und nicht nur über diejenigen, die zu dem Wert der aktuellen Dimension zählen. Die Dimensionen des Diagramms werden also nicht berücksichtigt. Mit **TOTAL [<fld {.fld}>]**, wobei auf den Zusatz **TOTAL** eine Liste aus mindestens einem Feldnamen (d. h. einer Teilmenge der Diagrammdimensionsvariablen) folgt, erstellen Sie eine Teilmenge aller möglichen Werte.

Beispiele und Ergebnisse:

Results table

SalesGroup	Amount	Concat(Team)	Concat(TOTAL <SalesGroup> Team)
East	25000	Alpha	AlphaBetaDeltaGammaGamma
East	20000	BetaGammaGamma	AlphaBetaDeltaGammaGamma
East	14000	Delta	AlphaBetaDeltaGammaGamma
West	17000	Epsilon	EpsilonEtaThetaZeta
West	14000	Eta	EpsilonEtaThetaZeta
West	23000	Theta	EpsilonEtaThetaZeta
West	19000	Zeta	EpsilonEtaThetaZeta

Funktionsbeispiele

Beispiel	Ergebnis
`Concat(Team)`	Die Tabelle wurde aus den Dimensionen SalesGroup und Amount sowie Variationen der Kennzahl Concat(Team) erstellt. Beachten Sie bitte: Selbst wenn sich bei Nichtbeachtung des Gesamtergebnisses die Daten für acht Werte von Team über zwei Werte von SalesGroup verteilen, ist das einzige Ergebnis der Kennzahl Concat(Team), das mehr als einen Stringwert Team in der Tabelle zusammenfasst, die Zeile mit der Kennzahl Amount 20000. Dadurch lautet das Ergebnis BetaGammaGamma. Der Grund hierfür ist, dass sich drei Werte für Amount 20000 in der Datenquelle befinden. Alle anderen Ergebnisse werden nicht zusammengefasst, wenn die Kennzahl über die Dimensionen aufgeteilt wird, da es nur einen Wert Team für jede Kombination von SalesGroup und Amount gibt.
`Concat (DISTINCT Team,', ')`	Beta, Gamma, weil der Zusatz DISTINCT bedeutet, dass das duplizierte Ergebnis Gamma nicht berücksichtigt wird. Außerdem ist als Trennzeichen ein Komma mit einem nachfolgenden Leerzeichen definiert.
`Concat (TOTAL <SalesGroup> Team)`	Alle Stringwerte für alle Werte von Team werden zusammengefasst, wenn der Zusatz TOTAL verwendet wird. Bei Festlegung der Feldauswahl <SalesGroup> werden die Ergebnisse in zwei Werte der Dimension SalesGroup unterteilt. Für SalesGroupEast lauten die Ergebnisse AlphaBetaDeltaGammaGamma. Für SalesGroupWest lauten die Ergebnisse EpsilonEtaThetaZeta.
`Concat (TOTAL <SalesGroup> Team,';', Amount)`	Durch Hinzufügen des Arguments für **sort-weight**: Amount werden die Ergebnisse nach dem Wert der Dimension Amount sortiert. Die Ergebnisse lauten DeltaBetaGammaGammaAlpha und EtaEpsilonZEtaTheta.

Im Beispiel verwendete Daten:

```
TeamData:
LOAD * inline [
SalesGroup|Team|Date|Amount
East|Gamma|01/05/2013|20000
East|Gamma|02/05/2013|20000
West|Zeta|01/06/2013|19000
East|Alpha|01/07/2013|25000
East|Delta|01/08/2013|14000
West|Epsilon|01/09/2013|17000
West|Eta|01/10/2013|14000
East|Beta|01/11/2013|20000
West|Theta|01/12/2013|23000
] (delimiter is '|');
```

FirstValue

FirstValue() liefert den Wert, der zuerst aus den durch die Formel definierten Datensätzen geladen wurde, sortiert nach der Bedingung **group by**.

Diese Funktion steht nur als Skriptfunktion zur Verfügung.

Syntax:

```
FirstValue ( expr)
```

Rückgabe Datentyp: dual

Argumente:

Argumente

Argument	Beschreibung
expr	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.

Beschränkungen:

Werden keine Textwerte gefunden, ist das Ergebnis NULL.

Beispiele und Ergebnisse:

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Fügen Sie einem Arbeitsblatt in Ihrer App dann die Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um das Ergebnis anzuzeigen.

Ergebnisdaten

Beispiel	Ergebnis	Ergebnisse in einem Arbeitsblatt
TeamData: LOAD * inline [SalesGroup|Team|Date|Amount East|Gamma|01/05/2013|20000 East|Gamma|02/05/2013|20000 West|Zeta|01/06/2013|19000 East|Alpha|01/07/2013|25000 East|Delta|01/08/2013|14000 West|Epsilon|01/09/2013|17000 West|Eta|01/10/2013|14000 East|Beta|01/11/2013|20000 West|Theta|01/12/2013|23000] (delimiter is '|'); FirstValue1: LOAD SalesGroup,FirstValue(Team) as FirstTeamLoaded Resident TeamData Group By SalesGroup;	SalesGroup East West	FirstTeamLoaded Gamma Zeta

LastValue

LastValue() liefert den Wert, der zuletzt aus den durch die Formel definierten Datensätzen geladen wurde, sortiert nach der Bedingung **group by**.

Diese Funktion steht nur als Skriptfunktion zur Verfügung.

Syntax:

```
LastValue ( expr )
```

Rückgabe Datentyp: dual

Argumente:

Argumente

Argument	Beschreibung
expr	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.

Beschränkungen:

Werden keine Textwerte gefunden, ist das Ergebnis NULL.

Beispiele und Ergebnisse:

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Fügen Sie einem Arbeitsblatt in unserer App dann mindestens diejenigen Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um das Ergebnis anzuzeigen.

Damit die Anzeige genauso wie in der unteren Ergebnisspalte aussieht, schalten Sie im Eigenschaftsfenster unter "Sortierung" von "Auto" auf "Benutzerdefiniert" um und heben Sie anschließend die numerische und alphabetische Sortierung auf.

Beispiel	Ergebnis	Ergebnis mit benutzerdefinierter Sortierung

`TeamData:`
`LOAD * inline [`
`SalesGroup|Team|Date|Amount`
`East|Gamma|01/05/2013|20000`
`East|Gamma|02/05/2013|20000`
`West|Zeta|01/06/2013|19000`
`East|Alpha|01/07/2013|25000`
`East|Delta|01/08/2013|14000`
`West|Epsilon|01/09/2013|17000`
`West|Eta|01/10/2013|14000`
`East|Beta|01/11/2013|20000`
`West|Theta|01/12/2013|23000`
`] (delimiter is '|');`
 	
`LastValue1:`
`LOAD SalesGroup,LastValue(Team) as LastTeamLoaded`
`Resident TeamData Group By SalesGroup;`
	SalesGroup
	East
	West
		LastTeamLoaded
		Beta
		Theta

MaxString

MaxString() sucht nach Stringwerten in der Formel und gibt den letzten Textwert in alphabetischer Reihenfolge über mehrere Datensätze hinweg zurück, wie in einer **group by**-Bedingung definiert.

Syntax:

```
MaxString ( expr )
```

Rückgabe Datentyp: dual

Argumente:

Argument	Beschreibung
expr	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.

Beschränkungen:

Werden keine Textwerte gefunden, ist das Ergebnis NULL.

Beispiele und Ergebnisse:

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Fügen Sie einem Arbeitsblatt in Ihrer App dann die Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um das Ergebnis anzuzeigen.

Beispiel	Ergebnis	
TeamData: LOAD * inline [SalesGroup|Team|Date|Amount East|Gamma|01/05/2013|20000 East|Gamma|02/05/2013|20000 West|Zeta|01/06/2013|19000 East|Alpha|01/07/2013|25000 East|Delta|01/08/2013|14000 West|Epsilon|01/09/2013|17000 West|Eta|01/10/2013|14000 East|Beta|01/11/2013|20000 West|Theta|01/12/2013|23000] (delimiter is '|'); Concat1: LOAD SalesGroup,MaxString(Team) as MaxString1 Resident TeamData Group By SalesGroup;	SalesGroup East West	MaxString1 Gamma Zeta

Beispiel	Ergebnis	
Vorausgesetzt, dass die **TeamData**-Tabelle wie im vorherigen Beispiel geladen wird und das Datenladeskript über den SET-Befehl verfügt: `SET DateFormat='DD/MM/YYYY';`: `LOAD SalesGroup,MaxString(Date) as MaxString2 Resident TeamData Group By SalesGroup;`	SalesGroup East West	MaxString2 01/11/2013 01/12/2013

MaxString - Diagrammfunktion

MaxString() sucht nach Stringwerten in der Formel oder im Feld und liefert den letzten Textwert in alphabetischer Reihenfolge.

Syntax:

```
MaxString({[SetExpression] [TOTAL [<fld{, fld}>]]} expr)
```

Rückgabe Datentyp: dual

Argumente:

Argumente

Argument	Beschreibung
expr	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.
SetExpression	Standardmäßig berechnet sich die Aggregierungsfunktion über alle wählbaren Werte. Alternativ können Sie die der Berechnung zugrunde liegenden Werte über die Auswahlformel bestimmen.
TOTAL	Der Zusatz **TOTAL** vor der Funktion bewirkt, dass die Berechnung über alle ausgewählten bzw. wählbaren Werte erfolgt, und nicht nur über diejenigen, die zu dem Wert der aktuellen Dimension zählen. Die Dimensionen des Diagramms werden also nicht berücksichtigt. Mit **TOTAL [<fld {.fld}>]**, wobei auf den Zusatz **TOTAL** eine Liste aus mindestens einem Feldnamen (d. h. einer Teilmenge der Diagrammdimensionsvariablen) folgt, erstellen Sie eine Teilmenge aller möglichen Werte.

Beschränkungen:

Wenn die Formel keine Werte mit String-Darstellung enthält, wird NULL ausgegeben.

Beispiele und Ergebnisse:

Ergebnistabelle

SalesGroup	Amount	MaxString(Team)	MaxString(Date)
East	14000	Delta	2013/08/01
East	20000	Gamma	2013/11/01
East	25000	Alpha	2013/07/01
West	14000	Eta	2013/10/01
West	17000	Epsilon	2013/09/01
West	19000	Zeta	2013/06/01
West	23000	Theta	2013/12/01

Funktionsbeispiele

Beispiel	Ergebnis
`MaxString (Team)`	Es gibt dreimal den Wert 20000 für die Dimension Amount: zwei von Gamma (an verschiedenen Tagen) und einen von Beta. Das Ergebnis dieser Kennzahl MaxString (Team) ist deshalb Gamma, weil es sich hierbei um den höchsten Wert der sortierten Strings handelt.
`MaxString (Date)`	2013/11/01 ist der späteste Date-Wert der drei mit der Dimension Amount assoziierten Werte. Dabei wird angenommen, dass der SET-Befehl `SET DateFormat='YYYY-MM-DD';'` im Skript verwendet wird.

Im Beispiel verwendete Daten:

```
TeamData:
LOAD * inline [
SalesGroup|Team|Date|Amount
East|Gamma|01/05/2013|20000
East|Gamma|02/05/2013|20000
West|Zeta|01/06/2013|19000
East|Alpha|01/07/2013|25000
East|Delta|01/08/2013|14000
West|Epsilon|01/09/2013|17000
West|Eta|01/10/2013|14000
East|Beta|01/11/2013|20000
West|Theta|01/12/2013|23000
] (delimiter is '|');
```

MinString

MinString() sucht nach Stringwerten in der Formel und gibt den ersten Textwert in alphabetischer Reihenfolge über mehrere Datensätze hinweg zurück, wie in einer **group by**-Bedingung definiert.

Syntax:

```
MinString ( expr )
```

Rückgabe Datentyp: dual

Argumente:

Argumente

Argument	Beschreibung
expr	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.

Beschränkungen:

Werden keine Textwerte gefunden, ist das Ergebnis NULL.

Beispiele und Ergebnisse:

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Fügen Sie einem Arbeitsblatt in Ihrer App dann die Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um das Ergebnis anzuzeigen.

Ergebnisdaten

Beispiel	Ergebnis	
TeamData: LOAD * inline [SalesGroup|Team|Date|Amount East|Gamma|01/05/2013|20000 East|Gamma|02/05/2013|20000 West|Zeta|01/06/2013|19000 East|Alpha|01/07/2013|25000 East|Delta|01/08/2013|14000 West|Epsilon|01/09/2013|17000 West|Eta|01/10/2013|14000 East|Beta|01/11/2013|20000 West|Theta|01/12/2013|23000] (delimiter is '|'); Concat1: LOAD SalesGroup,MinString(Team) as MinString1 Resident TeamData Group By SalesGroup;	SalesGroup East West	MinString1 Alpha Epsilon
Vorausgesetzt, dass die **TeamData**-Tabelle wie im vorherigen Beispiel geladen wird und das Datenladeskript über den SET-Befehl verfügt: SET DateFormat='DD/MM/YYYY';': LOAD SalesGroup,MinString(Date) as MinString2 Resident TeamData Group By SalesGroup;	SalesGroup East West	MinString2 01/05/2013 01/06/2013

MinString - Diagrammfunktion

MinString() sucht nach Stringwerten in der Formel oder im Feld und liefert den ersten Textwert in alphabetischer Reihenfolge.

Syntax:

```
MinString({[SetExpression] [TOTAL [<fld {, fld}>]]} expr)
```

Rückgabe Datentyp: dual

Argumente:

Argumente

Argument	Beschreibung
expr	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.
SetExpression	Standardmäßig berechnet sich die Aggregierungsfunktion über alle wählbaren Werte. Alternativ können Sie die der Berechnung zugrunde liegenden Werte über die Auswahlformel bestimmen.
TOTAL	Der Zusatz **TOTAL** vor der Funktion bewirkt, dass die Berechnung über alle ausgewählten bzw. wählbaren Werte erfolgt, und nicht nur über diejenigen, die zu dem Wert der aktuellen Dimension zählen. Die Dimensionen des Diagramms werden also nicht berücksichtigt. Mit **TOTAL [<fld {.fld}>]**, wobei auf den Zusatz **TOTAL** eine Liste aus mindestens einem Feldnamen (d. h. einer Teilmenge der Diagrammdimensionsvariablen) folgt, erstellen Sie eine Teilmenge aller möglichen Werte.

Beispiele und Ergebnisse:

Beispieldaten

SalesGroup	Amount	MinString(Team)	MinString(Date)
East	14000	Delta	2013/08/01
East	20000	Beta	2013/05/01
East	25000	Alpha	2013/07/01
West	14000	Eta	2013/10/01
West	17000	Epsilon	2013/09/01
West	19000	Zeta	2013/06/01
West	23000	Theta	2013/12/01

Funktionsbeispiele

Beispiele	Ergebnisse
`MinString (Team)`	Es gibt dreimal den Wert 20000 für die Dimension Amount: zwei von Gamma (an verschiedenen Tagen) und einen von Beta. Das Ergebnis dieser Kennzahl MinString (Team) ist deshalb Beta, weil es sich hierbei um den ersten Wert der sortierten Strings handelt.
`MinString (Date)`	2013/11/01 ist der früheste Date-Wert der drei mit der Dimension Amount assoziierten Werte. Dabei wird angenommen, dass der SET-Befehl `SET DateFormat='YYYY-MM-DD';'` im Skript verwendet wird.

Im Beispiel verwendete Daten:

```
TeamData:
LOAD * inline [
SalesGroup|Team|Date|Amount
East|Gamma|01/05/2013|20000
East|Gamma|02/05/2013|20000
West|Zeta|01/06/2013|19000
East|Alpha|01/07/2013|25000
East|Delta|01/08/2013|14000
West|Epsilon|01/09/2013|17000
West|Eta|01/10/2013|14000
East|Beta|01/11/2013|20000
West|Theta|01/12/2013|23000
] (delimiter is '|');
```

Funktionen für synthetische Dimensionen

Eine synthetische Dimension wird in der App aus Werten erstellt, die von den Funktionen für synthetische Dimensionen und nicht direkt aus Feldern im Datenmodell generiert wurden. Wenn Werte, die von einer Funktion für eine synthetische Dimension generiert wurden, in einem Diagramm als berechnete Dimension verwendet werden, wird dadurch eine synthetische Dimension erstellt. Synthetische Dimensionen ermöglichen Ihnen das Erstellen von beispielsweise Diagrammen und Dimensionen mit Werten aus Ihren Daten – also mit dynamischen Dimensionen.

Synthetische Dimensionen sind von Auswahlen nicht betroffen.

Die folgenden synthetischen Dimensionen können in Diagrammen verwendet werden.

ValueList

ValueList() liefert eine Liste der angegebenen Werte. Wird diese Funktion als dynamische Dimension verwendet, lassen sich synthetische Dimensionen generieren.

```
ValueList - Diagrammfunktion (v1 {, Expression})
```

ValueLoop

ValueLoop() liefert eine Liste der angegebenen Werte. Wird diese Funktion als dynamische Dimension verwendet, lassen sich synthetische Dimensionen generieren.

```
ValueLoop - Diagrammfunktion(from [, to [, step ]])
```

ValueList - Diagrammfunktion

ValueList() liefert eine Liste der angegebenen Werte. Wird diese Funktion als dynamische Dimension verwendet, lassen sich synthetische Dimensionen generieren.

*In Diagrammen mit einer synthetischen Dimension mit der **ValueList**-Funktion kann der Dimensionswert referenziert werden. Um einen Bezug zwischen Dimensions- und Formelwerten herzustellen, wird dieselbe **ValueList**-Funktion mit denselben Parametern auch in der Formel des Diagramms verwendet. Diese Funktion kann zwar überall im Layout verwendet werden, aber abgesehen von synthetischen Dimensionen ist ihre Verwendung nur innerhalb einer Aggregierungsfunktion sinnvoll.*

Synthetische Dimensionen sind von Auswahlen nicht betroffen.

Syntax:

```
ValueList(v1 {,...})
```

Rückgabe Datentyp: dual

Argumente:

Argumente

Argument	Beschreibung
v1	Statischer Wert (normalerweise ein String, kann aber auch eine Zahl sein).
{,...}	Optionale Liste statischer Werte.

Beispiele und Ergebnisse:

Funktionsbeispiele

Beispiel	Ergebnis
`ValueList ('Number of Orders', 'Average Order Size', 'Total Amount')`	Bei Verwendung zur Erstellung einer Dimension in einer Tabelle führt dies beispielsweise dazu, dass die drei Stringwerte als Zeilenbeschriftungen in der Tabelle eingesetzt werden. Diese können dann als Formel referenziert werden.

Beispiel	Ergebnis
=IF(ValueList ('Number of Orders', 'Average Order Size', 'Total Amount') = 'Number of Orders', count (SaleID), IF(ValueList ('Number of Orders', 'Average Order Size', 'Total Amount') = 'Average Order Size', avg (Amount), sum (Amount)))	Diese Formel zieht die Werte der erstellten Dimension heran und referenziert diese in einem verschachtelten IF-Befehl als Eingabe für drei Aggregierungsfunktionen: **ValueList()** Created dimension / Year / Added expression 522.00 Number of Orders 2012 5.00 Number of Orders 2013 7.00 Average Order Size 2012 13.20 Average Order Size 2013 15.43 Total Amount 2012 66.00 Total Amount 2013 108.00

In Beispielen verwendete Daten:

```
SalesPeople:
LOAD * INLINE [
SaleID|SalesPerson|Amount|Year
1|1|12|2013
2|1|23|2013
3|1|17|2013
4|2|9|2013
5|2|14|2013
6|2|29|2013
7|2|4|2013
8|1|15|2012
9|1|16|2012
10|2|11|2012
11|2|17|2012
12|2|7|2012
] (delimiter is '|');
```

ValueLoop - Diagrammfunktion

ValueLoop() liefert eine Liste der angegebenen Werte. Wird diese Funktion als dynamische Dimension verwendet, lassen sich synthetische Dimensionen generieren.
Die generierten Werte beginnen mit dem Wert **from** und enden mit dem Wert **to** einschließlich schrittweiser Zwischenwerte.

*In Diagrammen mit einer synthetischen Dimension mit der **ValueLoop**-Funktion kann der Dimensionswert referenziert werden. Um einen Bezug zwischen Dimensions- und Formelwerten herzustellen, wird dieselbe **ValueLoop**-Funktion mit denselben Parametern auch in der Formel des Diagramms verwendet. Diese Funktion kann zwar überall im Layout verwendet werden, aber abgesehen von synthetischen Dimensionen ist ihre Verwendung nur innerhalb einer Aggregierungsfunktion sinnvoll.*

Synthetische Dimensionen sind von Auswahlen nicht betroffen.

Syntax:

```
ValueLoop(from [, to [, step ]])
```

Rückgabe Datentyp: dual

Argumente:

Argumente

Argumente	Beschreibung
from	Startwert in der zu generierenden Gruppe an Werten.
to	Endwert in der zu generierenden Gruppe an Werten.
step	Schrittgröße des Zwischenwerts.

Beispiele und Ergebnisse:

Funktionsbeispiele

Beispiel	Ergebnis
`ValueLoop (1, 10)`	Dadurch wird eine Dimension in einer Tabelle erstellt, die beispielsweise für eine nummerierte Beschriftung eingesetzt werden kann. Das hier angegebene Beispiel ergibt eine Nummerierung der Werte von 1 bis 10. Diese Werte können dann als Formel referenziert werden.
`ValueLoop (2, 10,2)`	In diesem Beispiel ergibt das die Nummerierung als 2, 4, 6, 8 und 10, da das Argument step einen Wert von 2 besitzt.

Verschachtelte Aggregierung

Es können Situationen auftreten, in denen Sie eine Aggregierung auf das Ergebnis einer anderen Aggregierung anwenden müssen. Dies wird als verschachtelte Aggregierung bezeichnet.

In den meisten Diagrammformeln können keine Aggregierungen verschachtelt werden. Sie können aber Aggregierungen verschachteln, wenn Sie den Qualifizierer **TOTAL** in der inneren Aggregierungsfunktion verwenden.

Es sind bis zu 100 Verschachtelungsebenen erlaubt.

Verschachtelte Aggregierung mit dem Zusatz TOTAL

Beispiel:

Angenommen, Sie möchten die Summe des Feldes **Sales** berechnen, aber nur für die Aufträge mit einem **OrderDate** aus dem letzten Jahr. Das letzte Jahr lässt sich mithilfe der Aggregierungsfunktion **Max(TOTAL** `Year(OrderDate)`**)** berechnen.

Folgende Aggregierung führt zum gewünschten Ergebnis:

```
Sum(If(Year(OrderDate)=Max(TOTAL Year(OrderDate)), Sales))
```

Qlik Sense erfordert den Einschluss des Qualifizierers **TOTAL** für diese Art von Verschachtelung. Das ist für den gewünschten Vergleich erforderlich. Diese Art der Verschachtelung von Aggregierungsfunktionen kommt häufig vor und ist ein sinnvolles Verfahren.

Siehe auch:

Aggr - Diagrammfunktion (page 566)

8.3 Aggr - Diagrammfunktion

Aggr() berechnet eine Reihe von Werten für die Formel aggregiert über die angegebene Dimension oder Dimensionen. Beispielsweise den maximalen Umsatzwert pro Kunde oder Region.

Die **Aggr**-Funktion wird für verschachtelte Aggregierungen verwendet, bei denen der erste Parameter (die innere Aggregierung) einmal pro Dimensionswert berechnet wird. Die Dimensionen werden im zweiten Parameter (und in den folgenden Parametern) angegeben.

Zudem sollte die Funktion **Aggr** in einer äußeren Aggregierungsfunktion eingeschlossen sein. Dazu wird die Reihe von Ergebnissen der Funktion **Aggr** als Eingabe für die Aggregierung verwendet, in der sie verschachtelt ist.

Syntax:

```
Aggr({SetExpression}[DISTINCT] [NODISTINCT ] expr, StructuredParameter{,
StructuredParameter})
```

Rückgabe Datentyp: dual

Argumente:

Argumente

Argument	Beschreibung
expr	Eine aus einer Aggregierungsfunktion bestehende Formel. Standardmäßig berechnet sich die Aggregierungsfunktion über alle wählbaren Werte.
StructuredParameter	StructuredParameter besteht aus einer Dimension und optional aus Sortierkriterien im folgenden Format: `(Dimension(Sort-type, Ordering))` Die Dimension ist ein Einzelfeld und kann keine Formel sein. Mithilfe der Dimension wird die Reihe der Werte festgelegt, für die die Formel Aggr berechnet wird. Wenn Sortierkriterien enthalten sind, wird die Reihe der von der Aggr-Funktion erstellten und für die Dimension berechneten Werte sortiert. Dies ist wichtig, wenn die Sortierreihenfolge das Ergebnis der in der Aggr-Funktion eingeschlossenen Formel beeinflusst. Details zur Verwendung der Sortierkriterien finden Sie unter Hinzufügen von Sortierkriterien zur Dimension im strukturierten Parameter.
SetExpression	Standardmäßig berechnet sich die Aggregierungsfunktion über alle wählbaren Werte. Alternativ können Sie die der Berechnung zugrunde liegenden Werte über die Auswahlformel bestimmen.
DISTINCT	Wird dem Formelargument der Qualifizierer **distinct** vorangestellt oder wird gar kein Qualifizierer verwendet, generiert jede einzelne Kombination aus Dimensionswerten nur einen Rückgabewert. So werden Aggregierungen normalerweise verwendet – jede unterschiedliche Kombination aus Dimensionswerten ergibt eine Zeile im Diagramm.
NODISTINCT	Wird dem Formelargument der Qualifizierer **nodistinct** vorangestellt, kann jede Kombination aus Dimensionswerten abhängig von der zugrunde liegenden Datenstruktur mehrere Rückgabewerte generieren. Ist nur eine Dimension vorhanden, gibt die Funktion **aggr** ein Array mit so vielen Elementen zurück, wie Zeilen in den Quelldaten vorhanden sind.

Grundlegende Aggregierungsfunktionen wie **Sum**, **Min** und **Avg** geben einen einzelnen numerischen Wert zurück. Die Funktion Aggr() kann hingegen mit dem Erstellen einer temporären abgestuften Trefferliste (einer virtuellen Tabelle) verglichen werden, mit der eine weitere Aggregierung durchgeführt werden kann. Beispielsweise ergibt sich durch die Berechnung eines durchschnittlichen Verkaufswerts durch Addieren der Verkäufe pro Kunde in einer **Aggr()**-Anweisung und anschließender Berechnung des Durchschnitts der summierten Ergebnisse: **Avg(TOTAL Aggr(Sum(Sales),Customer))**.

Verwenden Sie die Funktion Aggr() in berechneten Dimensionen, wenn Sie verschachtelte Diagrammaggregierungen auf mehreren Ebenen erstellen möchten.

Beschränkungen:

Jede Dimension in einer Aggr()-Funktion muss ein einzelnes Feld sein und darf keine Formel (berechnete Dimension) sein.

Hinzufügen von Sortierkriterien zur Dimension im strukturierten Parameter

In seiner Grundform ist das Argument StructuredParameter in der Aggr-Funktionssyntax eine einzelne Dimension. Mit der Formel Aggr(Sum(Sales, Month)) wird der Gesamtumsatz für jeden Monat ermittelt. Bei Einschluss in einer anderen Aggregierungsfunktion kann es jedoch zu unerwarteten Ergebnissen kommen, sofern keine Sortierkriterien verwendet werden. Das liegt daran, dass manche Dimensionen numerisch, alphabetisch usw. sortiert werden können.

Im StructuredParameter-Argument in der Aggr-Funktion können Sie die Sortierkriterien für die Dimension in Ihrer Formel angeben. Auf diese Weise wenden Sie eine Sortierreihenfolge auf die virtuelle Tabelle an, die von der Aggr-Funktion erstellt wird.

Das Argument StructuredParameter weist die folgende Syntax auf:

```
(FieldName, (Sort-type, Ordering))
```

Strukturierte Parameter können verschachtelt werden:

```
(FieldName, (FieldName2, (Sort-type, Ordering)))
```

Der Sortiertyp kann NUMERIC, TEXT, FREQUENCY oder LOAD_ORDER sein.

Die mit jedem Ordnungstyp verknüpften Sortiertypen lauten wie folgt:

Zulässige Ordnungstypen

Sortiertyp	**Zulässige Ordnungstypen**
NUMERIC	ASCENDING, DESCENDING oder REVERSE
TEXT	ASCENDING, A2Z, DESCENDING, REVERSE oder Z2A
FREQUENCY	DESCENDING, REVERSE oder ASCENDING
LOAD_ORDER	ASCENDING, ORIGINAL, DESCENDING oder REVERSE

Die Ordnungstypen REVERSE und DESCENDING sind gleichwertig.

Für den Sortiertyp TEXT sind die Ordnungstypen ASCENDING und A2Z gleichwertig und DESCENDING, REVERSE und Z2A sind gleichwertig.

Für den Sortiertyp LOAD_ORDER sind die Ordnungstypen ASCENDING und ORIGINAL gleichwertig.

Beispiele: Diagrammformeln mit Aggr

Beispiele – Diagrammformeln

Diagrammformel – Beispiel 1

Ladeskript

Laden Sie die folgenden Daten als Inline-Ladevorgang in den Dateneditor, um das folgende Diagrammformelbeispiel zu erstellen.

```
ProductData:

LOAD * inline [
Customer|Product|UnitSales|UnitPrice
Astrida|AA|4|16
Astrida|AA|10|15
Astrida|BB|9|9
Betacab|BB|5|10
Betacab|CC|2|20
Betacab|DD|25|25
Canutility|AA|8|15
Canutility|CC|0|19
] (delimiter is '|');
```

Diagrammformel

Erstellen Sie eine KPI-Visualisierung in einem Qlik Sense Arbeitsblatt. Fügen Sie die folgende Formel als Kennzahl zum KPI hinzu:

```
Avg(Aggr(Sum(UnitSales*UnitPrice), Customer))
```

Ergebnis

376.7

Erläuterung

Die Formel `Aggr(Sum(UnitSales*UnitPrice), Customer)` findet den Gesamtwert der Verkäufe pro **Customer** und gibt eine Reihe von Werten zurück: 295, 715 und 120 für die drei **Customer**-Werte.

Wir haben die Werte temporär aufgelistet, ohne dass eine separate Tabelle oder Spalte mit diesen Werten erstellt wurde.

Diese Werte werden als Input für die **Avg()**-Funktion genutzt, um den durchschnittlichen Verkaufswert zu bestimmen: 376.7.

Diagrammformel – Beispiel 2

Ladeskript

Laden Sie die folgenden Daten als Inline-Ladevorgang in den Dateneditor, um das folgende Diagrammformelbeispiel zu erstellen.

```
ProductData:
```

```
LOAD * inline [
Customer|Product|UnitSales|UnitPrice
Astrida|AA|4|16
Astrida|AA|10|15
Astrida|BB|10|15
Astrida|BB|9|9
Betacab|BB|5|10
Betacab|BB|7|12
Betacab|CC|2|22
Betacab|CC|4|20
Betacab|DD|25|25
Canutility|AA|8|15
Canutility|AA|5|11
Canutility|CC|0|19
] (delimiter is '|');
```

Diagrammformel

Erstellen Sie eine Tabellenvisualisierung in einem Qlik Sense Arbeitsblatt mit **Customer**, **Product**, **UnitPrice** und **UnitSales** als Dimensionen. Fügen Sie die folgende Formel als Kennzahl zur Tabelle hinzu:

```
Aggr(NODISTINCT Max(UnitPrice), Customer, Product)
```

Ergebnis

Customer	Product	UnitPrice	UnitSales	Aggr(NODISTINCT Max(UnitPrice), Customer, Product)
Astrida	AA	15	10	16
Astrida	AA	16	4	16
Astrida	BB	9	9	15
Astrida	BB	15	10	15
Betacab	BB	10	5	12
Betacab	BB	12	7	12
Betacab	CC	20	4	22
Betacab	CC	22	2	22
Betacab	DD	25	25	25
Canutility	AA	11	5	15
Canutility	AA	15	8	15
Canutility	CC	19	0	19

Erläuterung

Ein Werte-Array: 16, 16, 15, 15, 12, 12, 22, 22, 25, 15, 15 und 19. Der Qualifizierer **nodistinct** bedeutet, dass die Werte ein Element für jede Zeile in den Quelldaten enthalten: jeder Wert ist der Maximalwert von **UnitPrice** für jeden **Customer** und jedes **Product**.

Diagrammformel – Beispiel 3

Ladeskript

Laden Sie die folgenden Daten als Inline-Ladevorgang in den Dateneditor, um das folgende Diagrammformelbeispiel zu erstellen.

```
Set vNumberOfOrders = 1000;

OrderLines:
Load
       RowNo() as OrderLineID,
       OrderID,
       OrderDate,
       Round((Year(OrderDate)-2005)*1000*Rand()*Rand()*Rand1) as Sales
       While Rand()<=0.5 or IterNo()=1;
Load * Where OrderDate<=Today();
Load
       Rand() as Rand1,
       Date(MakeDate(2013)+Floor((365*4+1)*Rand())) as OrderDate,
       RecNo() as OrderID
       Autogenerate vNumberOfOrders;

Calendar:
Load distinct
       Year(OrderDate) as Year,
       Month(OrderDate) as Month,
       OrderDate
       Resident OrderLines;
```

Diagrammformeln

Erstellen Sie eine Tabellenvisualisierung in einem Qlik Sense Arbeitsblatt mit **Year** und **Month** als Dimensionen. Fügen Sie die folgenden Formeln als Kennzahlen zur Tabelle hinzu:

- `Sum(Sales)`
- `Sum(Aggr( Rangesum(Above(Sum(Sales),0,12)), (Year, (Numeric, Ascending)), (Month, (Numeric, Ascending)) ))` in der Tabelle als „Structured Aggr()“ beschriftet.

Ergebnis

Year	Month	Sum(Sales)	Structured Aggr()
2013	Jan	53495	53495
2013	Feb	48580	102075

Year	Month	Sum(Sales)	Structured Aggr()
2013	Mar	25651	127726
2013	Apr	36585	164311
2013	May	61211	225522
2013	Jun	23689	249211
2013	Jul	42311	291522
2013	Aug	41913	333435
2013	Sep	28886	362361
2013	Oct	25977	388298
2013	Nov	44455	432753
2013	Dec	64144	496897
2014	Jan	67775	67775

Erläuterung

Dieses Beispiel zeigt die aggregierten Werte über einen Zwölf-Monats-Zeitraum für jedes Jahr in chronologischer aufsteigender Reihenfolge. Daher sind die strukturierten Parameter (Numerisch, Aufsteigend) Teil der Formel **Aggr()**. Zwei spezifische Dimensionen sind als strukturierte Parameter erforderlich: **Year** und **Month**, sortiert (1) **Year** (numerisch) und (2) **Month** (numerisch). Diese beiden Dimensionen müssen in der Tabellen- oder Diagrammvisualisierung verwendet werden. Das ist erforderlich, damit die Dimensionsliste der Funktion **Aggr()** mit den Dimensionen des in der Visualisierung verwendeten Objekts übereinstimmt.

Sie können den Unterschied zwischen diesen Kennzahlen in einer Tabelle oder in getrennten Liniendiagrammen vergleichen:

- `Sum(Aggr( Rangesum(Above(Sum(Sales),0,12)), (Year), (Month) ))`
- `Sum(Aggr( Rangesum(Above(Sum(Sales),0,12)), (Year, (Numeric, Ascending)), (Month, (Numeric, Ascending)) ))`

Es sollte klar zu sehen sein, dass nur die letztere Formel die gewünschte Akkumulierung der aggregierten Werte durchführt.

Siehe auch:

Einfache Aggregierungsfunktionen (page 341)

8.4 Farbfunktionen

Diese Diagrammfunktionen können in Formeln verwendet werden, die mit Einstellungen verknüpft sind, und bewerten die Farbeigenschaften von Diagrammobjekten sowie in Datenladeskripts.

*Qlik Sense unterstützt die Farbfunktionen **Color()**, **qliktechblue** und **qliktechgray** aus Gründen der Abwärtskompatibilität, ihre Verwendung wird aber nicht empfohlen.*

ARGB

ARGB() wird in Formeln verwendet, um die Farbeigenschaften eines Diagrammobjekts festzulegen oder auszuwerten. Die Farbe wird durch eine rote **r**, eine grüne **g** und eine blaue **b** Komponente sowie einen Alpha-Faktor (Deckung) von **alpha** definiert.

```
ARGB (alpha, r, g, b)
```

HSL

HSL() wird in Formeln verwendet, um die Farbeigenschaften eines Diagrammobjekts zu definieren oder zu bestimmen. Die Farbe wird durch die Werte **hue**, **saturation** und **luminosity** zwischen 0 und 1 bestimmt.

```
HSL  (hue, saturation, luminosity)
```

RGB

RGB() gibt eine Ganzzahl zurück, die dem Farbcode der von drei Parametern definierten Farbe entspricht: der roten Komponente r, der grünen Komponente g und der blauen Komponente b. Diese Komponenten müssen Ganzzahlwerte zwischen 0 und 255 aufweisen. Die Funktion kann in Formeln verwendet werden, um die Farbeigenschaften eines Diagrammobjekts festzulegen oder zu bewerten.

```
RGB  (r, g, b)
```

Colormix1

Colormix1() wird in Formeln verwendet, um eine ARGB-Farbrepräsentation aus einem zweifarbigen Farbverlauf zwischen farbe0 und farbe1 zu liefern, die durch einen Wert zwischen 0 und 1 definiert ist.

```
Colormix1 (Value , ColorZero , ColorOne)
```

Value ist eine reelle Zahl zwischen 0 und 1.

- Ist Value gleich 0, ist das Ergebnis ColorZero .
- Ist Value gleich 1, ist das Ergebnis ColorOne .
- Ist 0 < Value< 1, ist das Ergebnis ein entsprechender Zwischenton.

ColorZero ist eine RGB-Farbe, die den Beginn des Farbverlaufs definiert.

ColorOne ist eine RGB-Farbe, die das Ende des Farbverlaufs definiert.

Beispiel:

```
Colormix1(0.5, red(), blue())
```

liefert:

```
ARGB(255,64,0,64) (purple)
```

Colormix2

Colormix2() wird in Formeln verwendet, um eine ARGB-Farbrepräsentation, die durch einen Wert zwischen -1 und 1 definiert ist, aus einem zweifarbigen Farbverlauf zwischen -1 und 1 über eine optionale Zwischenfarbe für die Mitte (0) des Farbverlaufs zu liefern.

```
Colormix2 (Value ,ColorMinusOne , ColorOne[ , ColorZero])
```

Value ist eine reelle Zahl zwischen -1 und 1.

- Ist Value gleich -1, ist das Ergebnis die erste Farbe.
- Ist Value gleich 1, ist das Ergebnis die zweite Farbe.
- Ist -1 < Value< 1, ist das Ergebnis die entsprechende Farbmischung.

ColorMinusOne ist eine RGB-Farbe, die den Beginn des Farbverlaufs definiert.

ColorOne ist eine RGB-Farbe, die das Ende des Farbverlaufs definiert.

ColorZero ist eine RGB-Farbe, die die Mitte des Farbverlaufs definiert.

SysColor

SysColor() liefert die n-te vom Windows-System benutzte ARGB-Farbe nr, wobei nr dem Parameter für die Windows API-Funktion **GetSysColor(nr)** entspricht.

```
SysColor (nr)
```

ColorMapHue

ColorMapHue() liefert eine ARGB-Farbrepräsentation einer Farbe aus einer Farbkarte, die die Farbton-Komponente des HSV-Farbmodells ist. Die Farbkarte beginnt bei Rot, durchläuft Gelb, Grün, Cyan, Blau sowie Magenta und kehrt zu Rot zurück. Für x ist eine Zahl zwischen 0 und 1 einzugeben.

```
ColorMapHue (x)
```

ColorMapJet

ColorMapJet() liefert eine ARGB-Farbrepräsentation einer Farbe aus einer Farbkarte, die bei Blau beginnt, Cyan, Gelb und Orange durchläuft und zu Rot zurückkehrt. Für x ist eine Zahl zwischen 0 und 1 einzugeben.

```
ColorMapJet (x)
```

Vordefinierte Farbfunktionen

Die folgenden Funktionen können in Formeln für vordefinierte Farben verwendet werden. Jede Funktion liefert eine RGB-Farbrepräsentation.

Optional kann ein Parameter für den Alpha-Faktor angegeben werden, in diesem Fall ergibt sich eine ARGB-Farbrepräsentation. Ein Alpha-Faktor von 0 entspricht vollständiger Transparenz und 255 vollständiger Deckung. Wenn für Alpha kein Wert eingegeben wird, wird von 255 ausgegangen.

Vordefinierte Farbfunktionen

Farbfunktion	RGB Wert
black ([alpha])	(0,0,0)

blue([alpha])	(0,0,128)
brown([alpha])	(128,128,0)
cyan([alpha])	(0,128,128)
darkgray([alpha])	(128,128,128)
green([alpha])	(0,128,0)
lightblue([alpha])	(0,0,255)
lightcyan([alpha])	(0,255,255)
lightgray([alpha])	(192,192,192)
lightgreen([alpha])	(0,255,0)
lightmagenta([alpha])	(255,0,255)
lightred([alpha])	(255,0,0)
magenta([alpha])	(128,0,128)
red([alpha])	(128,0,0)
white([alpha])	(255,255,255)
yellow([alpha])	(255,255,0)

Beispiele und Ergebnisse:

Beispiele und Ergebnisse

Beispiele	**Ergebnisse**
`Blue()`	RGB(0,0,128)
`Blue(128)`	ARGB(128,0,0,128)

ARGB

ARGB() wird in Formeln verwendet, um die Farbeigenschaften eines Diagrammobjekts festzulegen oder auszuwerten. Die Farbe wird durch eine rote **r**, eine grüne **g** und eine blaue **b** Komponente sowie einen Alpha-Faktor (Deckung) von **alpha** definiert.

Syntax:

```
ARGB(alpha, r, g, b)
```

Rückgabe Datentyp: dual

Argumente:

Argumente

Argument	Beschreibung
alpha	Transparenzwert im Bereich 0 - 255. 0 entspricht vollständiger Transparenz und 255 vollständiger Deckung.
r, g, b	Komponentenwerte für Rot, Grün und Blau. Eine Farbkomponente von 0 entspricht keinem Beitrag, eine Farbkomponente von 255 einem vollständigen Beitrag.

Alle Argumente müssen Formeln sein, die ganze Zahlen zwischen 0 und 255 ergeben.

Wird die numerische Komponente interpretiert und in Hexadezimalnotation formatiert, lassen sich die Werte der Farbkomponenten einfacher erkennen. Hellgrün hat beispielsweise den Wert 4 278 255 360, was FF00FF00 in hexadezimaler Schreibweise entspricht. Die ersten beiden Positionen 'FF' (255) kennzeichnen den **Alpha**-Kanal. Die nächsten beiden Positionen '00' stehen für den **Rot**-Anteil, die nächsten beiden Positionen 'FF' stehen für den **Grün**-Anteil und die letzten beiden Positionen '00' stehen für den **Blau**-Anteil.

RGB

RGB() gibt eine Ganzzahl zurück, die dem Farbcode der von drei Parametern definierten Farbe entspricht: der roten Komponente r, der grünen Komponente g und der blauen Komponente b. Diese Komponenten müssen Ganzzahlwerte zwischen 0 und 255 aufweisen. Die Funktion kann in Formeln verwendet werden, um die Farbeigenschaften eines Diagrammobjekts festzulegen oder zu bewerten.

Syntax:

```
RGB (r, g, b)
```

Rückgabe Datentyp: dual

Argumente:

Argumente

Argument	Beschreibung
r, g, b	Komponentenwerte für Rot, Grün und Blau. Eine Farbkomponente von 0 entspricht keinem Beitrag, eine Farbkomponente von 255 einem vollständigen Beitrag.

Alle Argumente müssen Formeln sein, die ganze Zahlen zwischen 0 und 255 ergeben.

Wird die numerische Komponente interpretiert und in Hexadezimalnotation formatiert, lassen sich die Werte der Farbkomponenten einfacher erkennen. Hellgrün hat beispielsweise den Wert 4 278 255 360, was FF00FF00 in hexadezimaler Schreibweise entspricht. Die ersten beiden Positionen 'FF' (255) kennzeichnen den **Alpha**-Kanal. In den Funktionen **RGB** und **HSL** ist dies stets 'FF' (deckend). Die nächsten beiden Positionen '00' stehen für den **Rot**-Anteil, die nächsten beiden Positionen 'FF' stehen für den **Grün**-Anteil und die letzten beiden Positionen '00' stehen für den **Blau**-Anteil.

Beispiel: Diagrammformel
In diesem Beispiel wird eine benutzerdefinierte Farbe auf ein Diagramm angewendet:

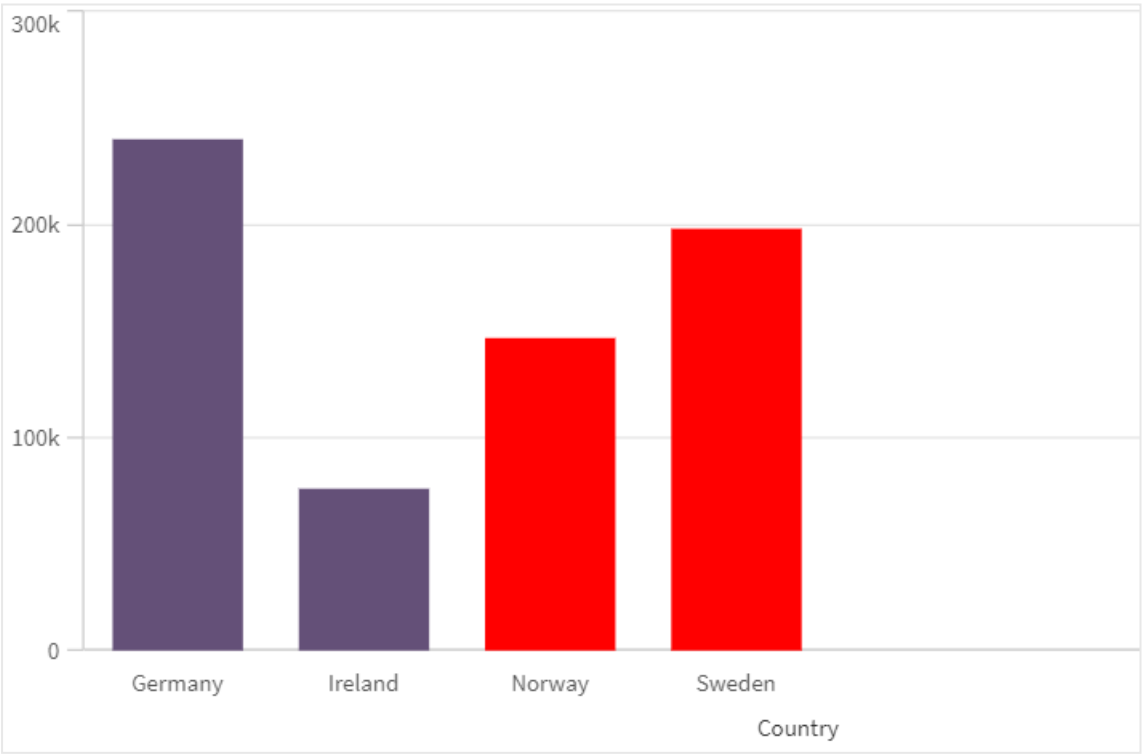
In diesem Beispiel verwendete Daten:

```
ProductSales:
Load * Inline
[Country,Sales,Budget
Sweden,100000,50000
Germany, 125000, 175000
Norway, 74850, 68500
Ireland, 45000, 48000
Sweden,98000,50000
Germany, 115000, 175000
Norway, 71850, 68500
Ireland, 31000, 48000
] (delimiter is ',');
```

Geben Sie die folgende Formel in das Eigenschaftsfenster **Farben und Legenden** ein:

```
If (Sum(Sales)>Sum(Budget),RGB(255,0,0),RGB(100,80,120))
```

Ergebnis:



Beispiel: Ladeskript

Das folgende Beispiel zeigt die äquivalenten RGB-Werte für Werte im hexadezimalen Format:

```
Load
Text(R & G & B) as Text,
RGB(R,G,B)      as Color;
Load
Num#(R,'(HEX)') as R,
Num#(G,'(HEX)') as G,
Num#(B,'(HEX)') as B
Inline
[R,G,B
01,02,03
AA,BB,CC];
```

Ergebnis:

Text	Farbe
010203	RGB(1,2,3)
AABBCC	RGB(170,187,204)

HSL

HSL() wird in Formeln verwendet, um die Farbeigenschaften eines Diagrammobjekts zu definieren oder zu bestimmen. Die Farbe wird durch die Werte **hue**, **saturation** und **luminosity** zwischen 0 und 1 bestimmt.

Syntax:

```
HSL (hue, saturation, luminosity)
```

Rückgabe Datentyp: dual

Argumente:

Argumente

Argument	Beschreibung
hue, saturation, luminosity	Die Werte für die Komponenten hue, saturation und luminosity liegen zwischen 0 und 1.

Alle Argumente müssen Formeln sein, die ganze Zahlen zwischen 0 und 1 ergeben.

Wird die numerische Komponente interpretiert und in Hexadezimalnotation formatiert, lassen sich die Werte der RGB -Farbkomponenten einfacher erkennen. Hellgrün hat beispielsweise den Wert 4 278 255 360, was FF00FF00 und RGB (0,255,0) in hexadezimaler Schreibweise entspricht. Dies entspricht HSL (80/240, 240/240, 120/240) – ein HSL -Wert von (0.33, 1, 0.5).

8.5 Konditionalfunktionen

Die Konditionalfunktionen berechnen alle eine Bedingung und liefern je nach Bedingungswert unterschiedliche Antworten. Die Funktionen können sowohl im Datenladeskript als auch in den Diagrammformeln verwendet werden.

Konditionalfunktionen – Übersicht

Jede Funktion wird nach der Übersicht genauer beschrieben. Sie können auch auf den Funktionsnamen in der Syntax klicken, um direkt auf die Details zu der spezifischen Funktion zuzugreifen.

alt

Die **alt**-Funktion liefert als Ergebnis das erste Argument, dem ein numerischer Wert zugeordnet werden kann. Wird kein solches gefunden, wird der letzte Wert ausgegeben. Es kann eine beliebige Anzahl von Argumenten benutzt werden.

```
alt (expr1[ , expr2 , expr3 , ...] , else)
```

class

Die **class**-Funktion weist den ersten Parameter einem Klassenintervall zu. Das Ergebnis ist ein dualer Wert in der Form von a<=x<b als Textwert, wobei a und b die obere und unteren Grenze des Intervalls sind, und der Startwert als numerischer Wert.

```
class (expression, interval [ , label [ , offset ]])
```

coalesce

Die Funktion **coalesce** liefert als Ergebnis den ersten der Parameter, die eine gültige non-NULL-Darstellung aufweisen. Es kann eine beliebige Anzahl von Argumenten benutzt werden.

```
coalesce(expr1[ , expr2 , expr3 , ...])
```

if

Die **if**-Funktion liefert einen Wert abhängig davon, ob die bereitgestellte Bedingung mit der Funktion True oder False ergibt.

```
if (condition , then , else)
```

match

Die **match**-Funktion vergleicht den ersten Parameter mit allen folgenden und liefert den numerischen Speicherort der übereinstimmenden Formeln. Beim Vergleich wird die Groß- und Kleinschreibung berücksichtigt.

```
match ( str, expr1 [ , expr2,...exprN ])
```

mixmatch

Die **mixmatch**-Funktion vergleicht den ersten Parameter mit allen folgenden und liefert den numerischen Speicherort der übereinstimmenden Formeln. Beim Vergleich spielt die Groß- und Kleinschreibung keine Rolle.

```
mixmatch ( str, expr1 [ , expr2,...exprN ])
```

pick
Die pick-Funktion liefert die *n*-te Formel in der Liste.

```
pick (n, expr1[ , expr2,...exprN])
```

wildmatch
Die **wildmatch**-Funktion vergleicht den ersten Parameter mit allen folgenden und liefert die übereinstimmenden Formeln. Sie lässt die Verwendung von Platzhalterzeichen (***** und **?**) in den Vergleichsstrings zu. ***** entspricht jeder Abfolge von Zeichen. **?** entspricht einem einzelnen Zeichen. Beim Vergleich wird die Groß- und Kleinschreibung berücksichtigt.

```
wildmatch ( str, expr1 [ , expr2,...exprN ])
```

alt

Die **alt**-Funktion liefert als Ergebnis das erste Argument, dem ein numerischer Wert zugeordnet werden kann. Wird kein solches gefunden, wird der letzte Wert ausgegeben. Es kann eine beliebige Anzahl von Argumenten benutzt werden.

Syntax:

```
alt(expr1[ , expr2 , expr3 , ...] , else)
```

Argumente:

Argumente

Argument	Beschreibung
expr1	Der erste Ausdruck prüft die Darstellung einer gültigen Zahl.
expr2	Der zweite Ausdruck prüft die Darstellung einer gültigen Zahl.
expr3	Der dritte Ausdruck prüft die Darstellung einer gültigen Zahl.
else	Der zurückzugebende Wert, wenn keiner der vorstehenden Parameter eine gültige Zahlendarstellung hat.

Die alt -Funktion wird häufig mit Zahlen- oder Datums-Interpretationsfunktionen verwendet. So kann Qlik Sense verschiedene Datumsformate nach Priorität testen. Sie kann in numerischen Formeln auch zur Handhabung von NULL-Werten verwendet werden.

Beispiele:

Beispiele

Beispiel	Ergebnis
`alt( date#( dat , 'YYYY/MM/DD' ), date#( dat , 'MM/DD/YYYY' ), date#( dat , 'MM/DD/YY' ), 'No valid date' )`	Diese Formel testet, ob das Feld „date“ ein Datum enthält, das einem der drei angegebenen Datumsformate entspricht. Ist dies der Fall, werden ein dualer Wert mit dem Original-String und die zugehörige Zahl für ein Datum ausgegeben. Anderenfalls wird der Text 'No valid date' ausgegeben (ohne gültigen numerischen Wert).
`alt(Sales,0) + alt(Margin,0)`	Diese Formel fügt die Felder Sales und Margin hinzu, wodurch fehlende Werte (NULL) mit 0 ersetzt werden.

class

Die **class**-Funktion weist den ersten Parameter einem Klassenintervall zu. Das Ergebnis ist ein dualer Wert in der Form von a<=x<b als Textwert, wobei a und b die obere und unteren Grenze des Intervalls sind, und der Startwert als numerischer Wert.

Syntax:

```
class(expression, interval [ , label [ , offset ]])
```

Argumente:

Argumente

Argument	Beschreibung
interval	Eine Zahl, welche die Intervallgröße angibt.
label	Ein beliebiger String, der 'x' im Ergebnistext ersetzen kann.
offset	Eine Zahl für den Abstand vom Standard-Startpunkt der Klassifizierung. Der Standard-Startwert ist in der Regel 0.

Beispiele:

Beispiele

Beispiel	Ergebnis
`class( var,10 )` mit `var = 23`	liefert '20<=x<30'
`class( var,5,'value' )` mit `var = 23`	liefert '20<= value <25'
`class( var,10,'x',5 )` mit `var = 23`	liefert '15<=x<25'

Beispiel: Ladeskript mit Verwendung von class

Beispiel: Ladeskript

Ladeskript

In diesem Beispiel wird eine Tabelle mit Namen und Alter von Personen geladen. Es soll ein Feld hinzugefügt werden, dass jede Person nach Altersgruppen in Zehn-Jahres-Intervallen klassifiziert. Die ursprüngliche Quelltabelle sieht wie folgt aus.

Ergebnisse

Name	Age
John	25
Karen	42
Yoshi	53

Sie können das Klassifizierungsfelds für die Altersgruppe hinzufügen, indem Sie einen vorangehenden Load-Befehl mithilfe der **class**-Funktion hinzufügen.

Erstellen Sie eine neue Registerkarte im Dateneditor und laden Sie die folgenden Daten als Inline-Ladevorgang: Erstellen Sie die Tabelle unten in Qlik Sense, um die Ergebnisse anzuzeigen.

```
LOAD *,
class(Age, 10, 'age') As Agegroup;

LOAD * INLINE
[ Age, Name
25, John
42, Karen
53, Yoshi];
```

Ergebnisse

Ergebnisse

Name	Age	Agegroup
John	25	20 <= age < 30
Karen	42	40 <= age < 50
Yoshi	53	50 <= age < 60

coalesce

Die Funktion **coalesce** liefert als Ergebnis den ersten der Parameter, die eine gültige non-NULL-Darstellung aufweisen. Es kann eine beliebige Anzahl von Argumenten benutzt werden.

Syntax:

```
coalesce(expr1[ , expr2 , expr3 , ...])
```

Argumente:

Argumente

Argument	Beschreibung
expr1	Der erste Ausdruck prüft das Vorliegen einer gültigen Nicht-NULL-Darstellung.
expr2	Der zweite Ausdruck prüft das Vorliegen einer gültigen Nicht-NULL-Darstellung.
expr3	Der dritte Ausdruck prüft das Vorliegen einer gültigen Nicht-NULL-Darstellung.

Beispiele:

Beispiele

Beispiel	Ergebnis
	Diese Formel ändert alle NULL-Werte eines Felds in „N/A“.
`Coalesce(ProductDescription, ProductName, ProductCode, 'no description available')`	Diese Formel wählt zwischen drei verschiedenen Produktbeschreibungsfeldern für den Fall, dass einige Felder keine Werte für das Produkt enthalten. Das erste der Felder in der angegebenen Reihenfolge mit einem Nicht-Null-Wert wird zurückgegeben. Wenn keines der Felder einen Wert enthält, ist das Ergebnis „keine Beschreibung verfügbar“.
`Coalesce(TextBetween(FileName, '"', '"'), FileName)`	Diese Formel entfernt etwaige umschließende Anführungszeichen aus dem Feld *FileName*. Wenn der angegebene *FileName* von Anführungszeichen umschlossen ist, werden diese entfernt, und der umschlossene *FileName* wird ohne Anführungszeichen zurückgegeben. Wenn die Funktion *TextBetween* keine Trennzeichen findet, gibt sie null zurück, was von **Coalesce** abgelehnt wird. Stattdessen wird der unformatierte *FileName* zurückgegeben.

if

Die **if**-Funktion liefert einen Wert abhängig davon, ob die bereitgestellte Bedingung mit der Funktion True oder False ergibt.

Syntax:

```
if(condition , then [, else])
```

Argumente

Argument	Beschreibung
condition	Formel, die logisch interpretiert wird.

Argument	Beschreibung
then	Der Typ der Formel kann beliebig gewählt werden. Ist *condition* True, liefert die if-Funktion den Wert von *then*.
else	Der Typ der Formel kann beliebig gewählt werden. Ist *condition* False, liefert die if-Funktion den Wert von *else*. Dieser Parameter ist optional. Wenn die *condition* False ist, wird NULL zurückgegeben, wenn Sie nicht else angegeben haben.

Beispiel

Beispiel	Ergebnis
`if( Amount>= 0, 'OK', 'Alarm' )`	Diese Formel überprüft, ob eine positive Zahl (0 oder größer) vorliegt, und gibt 'OK' zurück, wenn dies der Fall ist. Ist die Zahl weniger als 0, wird 'Alarm' zurückgegeben.

Beispiel: Ladeskript mit Verwendung von if

Beispiel: Ladeskript

Ladeskript

If kann mit anderen Methoden und Objekten, einschließlich Variablen, in einem Ladeskript verwendet werden. Wenn Sie z. B. eine *threshold*-Variable festlegen und anhand dieses Schwellenwerts ein Feld in das Datenmodell aufnehmen möchten, können Sie folgendermaßen vorgehen.

Erstellen Sie eine neue Registerkarte im Dateneditor und laden Sie die folgenden Daten als Inline-Ladevorgang: Erstellen Sie die Tabelle unten in Qlik Sense, um die Ergebnisse anzuzeigen.

```
Transactions:
Load * Inline [
transaction_id, transaction_date, transaction_amount, transaction_quantity, customer_id, size,
color_code
3750, 20180830, 23.56, 2, 2038593, L, Red
3751, 20180907, 556.31, 6, 203521, m, orange
3752, 20180916, 5.75, 1, 5646471, S, blue
3753, 20180922, 125.00, 7, 3036491, l, Black
3754, 20180922, 484.21, 13, 049681, xs, Red
3756, 20180922, 59.18, 2, 2038593, M, Blue
3757, 20180923, 177.42, 21, 203521, XL, Black
];

set threshold = 100;

/* Create new table called Transaction_Buckets
Compare transaction_amount field from Transaction table to threshold of 100.
Output results into a new field called Compared to Threshold
 */

Transaction_Buckets:
```

```
Load
   transaction_id,
   If(transaction_amount > $(threshold),'Greater than $(threshold)','Less than $(threshold)')
as [Compared to Threshold]
Resident Transactions;
```

Ergebnisse

Qlik Sense Tabelle mit der Ausgabe nach Verwendung der *if*-Funktion im Ladeskript.

transaction_id	Verglichen mit Schwellenwert
3750	Kleiner als 100
3751	Größer als 100
3752	Kleiner als 100
3753	Größer als 100
3754	Größer als 100
3756	Kleiner als 100
3757	Größer als 100

Beispiele: Diagrammformeln mit Verwendung von if

Beispiele: Diagrammformeln

Diagrammformel 1

Ladeskript

Erstellen Sie eine neue Registerkarte im Dateneditor und laden Sie die folgenden Daten als Inline-Ladevorgang: Erstellen Sie nach dem Laden der Daten die Diagrammausdruckbeispiele unten in einer Qlik Sense-Tabelle.

```
MyTable:
LOAD * inline [Date, Location, Incidents
1/3/2016, Beijing, 0
1/3/2016, Boston, 12
1/3/2016, Stockholm, 3
1/3/2016, Toronto, 0
1/4/2016, Beijing, 0
1/4/2016, Boston, 8];
```

Qlik Sense Tabelle mit Beispielen der *if*-Funktion in einer Diagrammformel.

Datum	Adresse	Incidents	if(Incidents>=10, 'Critical', 'Ok')	if(Incidents>=10, 'Critical', If(Incidents>=1 and Incidents<10, 'Warning', 'Ok'))
1/3/2016	Beijing	0	Ok	Ok
1/3/2016	Boston	12	Critical	Critical
1/3/2016	Stockholm	3	Ok	Warning
1/3/2016	Toronto	0	Ok	Ok
1/4/2016	Beijing	0	Ok	Ok
1/4/2016	Boston	8	Ok	Warnung

Diagrammformel 2

Fügen Sie in einer neuen App das folgende Skript auf einer neuen Registerkarte im Dateneditor hinzu und laden Sie dann die Daten. Anschließend können Sie die Tabelle mit den Diagrammformeln unten erstellen.

```
SET FirstWeekDay=0;
Load
Date(MakeDate(2022)+RecNo()-1) as Date
Autogenerate 14;
```

Qlik Sense-Tabelle mit einem Beispiel der *if*-Funktion in einer Diagrammformel.

Datum	WeekDay(Date)	If(WeekDay (Date)>=5,'WeekEnd','Normal Day')
1/1/2022	Sa	WeekEnd
1/2/2022	So	WeekEnd
1/3/2022	Mo	Normal Day
1/4/2022	Di	Normal Day
1/5/2022	Mi	Normal Day
1/6/2022	Do	Normal Day
1/7/2022	Fr	Normal Day
1/8/2022	Sa	WeekEnd
1/9/2022	So	WeekEnd
1/10/2022	Mo	Normal Day
1/11/2022	Di	Normal Day
1/12/2022	Mi	Normal Day
1/13/2022	Do	Normal Day

Datum	WeekDay(Date)	If(WeekDay (Date)>=5,'WeekEnd','Normal Day')
1/14/2022	Fr	Normal Day

match

Die **match**-Funktion vergleicht den ersten Parameter mit allen folgenden und liefert den numerischen Speicherort der übereinstimmenden Formeln. Beim Vergleich wird die Groß- und Kleinschreibung berücksichtigt.

Syntax:

```
match( str, expr1 [ , expr2,...exprN ])
```

*Wenn die Groß- und Kleinschreibung beim Vergleich keine Rolle spielen soll, verwenden Sie die **mixmatch**-Funktion. Wenn die Groß- und Kleinschreibung beim Vergleich keine Rolle spielen soll und Wildcards eingesetzt werden sollen, verwenden Sie die **wildmatch**-Funktion.*

Beispiel: Ladeskript mit Verwendung von match

Beispiel: Ladeskript

Ladeskript

Sie können match zum Laden einer Teilmenge von Daten verwenden. Sie können zum Beispiel einen numerischen Wert für eine Formel in der Funktion liefern. Dann können Sie die geladenen Daten basierend auf dem numerischen Wert beschränken. Match liefert 0, wenn keine Übereinstimmung vorliegt. Alle Formeln, für die in diesem Beispiel keine Übereinstimmung vorliegt, liefern daher 0 und werden aus dem Datenladen mit dem WHERE-Befehl ausgeschlossen.

Erstellen Sie eine neue Registerkarte im Dateneditor und laden Sie die folgenden Daten als Inline-Ladevorgang: Erstellen Sie die Tabelle unten in Qlik Sense, um die Ergebnisse anzuzeigen.

```
Transactions:
Load * Inline [
transaction_id, transaction_date, transaction_amount, transaction_quantity, customer_id, size,
color_code
3750, 20180830, 23.56, 2, 2038593, L, Red
3751, 20180907, 556.31, 6, 203521, m, orange
3752, 20180916, 5.75, 1, 5646471, S, blue
3753, 20180922, 125.00, 7, 3036491, l, Black
3754, 20180922, 484.21, 13, 049681, xs, Red
3756, 20180922, 59.18, 2, 2038593, M, Blue
3757, 20180923, 177.42, 21, 203521, XL, Black
];

/*
Create new table called Transaction_Buckets
```

```
Create new fields called Customer, and Color code - Blue and Black
Load Transactions table.
Match returns 1 for 'Blue', 2 for 'Black'.
Does not return a value for 'blue' because match is case sensitive.
Only values that returned numeric value greater than 0
are loaded by WHERE statment into Transactions_Buckets table.
*/

Transaction_Buckets:
Load
 customer_id,
 customer_id as [Customer],
 color_code as [Color Code Blue and Black]
Resident Transactions
Where match(color_code,'Blue','Black') > 0;
```

Ergebnisse

Qlik Sense Tabelle mit der Ausgabe der match-Funktion im Ladeskript

Color Code Blue and Black	Customer
Black	203521
Black	3036491
Blue	2038593

Beispiele: Diagrammformeln mit Verwendung von match

Beispiele: Diagrammformeln

Diagrammformel 1

Ladeskript

Erstellen Sie eine neue Registerkarte im Dateneditor und laden Sie die folgenden Daten als Inline-Ladevorgang: Erstellen Sie nach dem Laden der Daten die Diagrammausdruckbeispiele unten in einer Qlik Sense-Tabelle.

```
MyTable:
Load * inline [Cities, Count
Toronto, 123
Toronto, 234
Toronto, 231
Boston, 32
Boston, 23
Boston, 1341
Beijing, 234
Beijing, 45
Beijing, 235
Stockholm, 938
```

```
Stockholm, 39
Stockholm, 189
zurich, 2342
zurich, 9033
zurich, 0039];
```

Die erste Formel in der Tabelle unten liefert 0 für „Stockholm“, da „Stockholm“ nicht in der Liste der Formeln in der **match**-Funktion enthalten ist. Es wird 0 für „Zurich“ geliefert, weil der **match**-Vergleich die Groß- und Kleinschreibung berücksichtigt.

Qlik Sense Tabelle mit Beispielen der *match*-Funktion in einer Diagrammformel.

Cities	**match(Cities,'Toronto','Boston','Beijing','Zurich')**	**match(Cities,'Toronto','Boston','Beijing','Stockholm','Zurich')**
Beijing	3	3
Boston	2	2
Stockholm	0	4
Toronto	1	1
zurich	0	5

Diagrammformel 2

Sie können match verwenden, um eine benutzerdefinierte Sortierung für eine Formel auszuführen.

Standardmäßig werden Spalten abhängig von den Daten numerisch oder alphabetisch sortiert.

Qlik Sense Tabelle mit einem Beispiel für die Standardsortierrreihenfolge

Cities
Beijing
Boston
Stockholm
Toronto
zurich

Gehen Sie wie folgt vor, um die Reihenfolge zu ändern:

1. Öffnen Sie den Abschnitt **Sortieren** für Ihr Diagramm im Fenster **Eigenschaften**.
2. Deaktivieren Sie die automatische Sortierung für die Spalte, in der Sie eine benutzerdefinierte Sortierung durchführen möchten.
3. Heben Sie die Auswahl von **Numerisch sortieren** und **Alphabetisch sortieren** auf.
4. Wählen Sie **Nach Formel sortieren** und geben Sie dann eine Formel wie die Folgende ein:
   `=match( Cities, 'Toronto','Boston','Beijing','Stockholm','zurich')`
   Die Sortierreihenfolge der Spalte Cities ändert sich.

Qlik Sense Tabelle mit einem Beispiel für das Ändern der Sortierreihenfolge mit der *match*-Funktion

Cities
Toronto
Boston
Beijing
Stockholm
zurich

Sie können auch den gelieferten numerischen Wert anzeigen.

Qlik Sense Tabelle mit einem Beispiel für numerische Werte, die von der *match*-Funktion geliefert werden

Cities	**Cities & ' - ' & match (Cities, 'Toronto','Boston', 'Beijing','Stockholm','zurich')**
Toronto	Toronto - 1
Boston	Boston - 2
Beijing	Beijing - 3
Stockholm	Stockholm - 4
zurich	zurich - 5

mixmatch

Die **mixmatch**-Funktion vergleicht den ersten Parameter mit allen folgenden und liefert den numerischen Speicherort der übereinstimmenden Formeln. Beim Vergleich spielt die Groß- und Kleinschreibung keine Rolle.

Syntax:

```
mixmatch( str, expr1 [ , expr2,...exprN ])
```

Wenn dagegen die Groß- und Kleinschreibung beim Vergleich eine Rolle spielen soll, verwenden Sie die **match**-Funktion. Wenn die Groß- und Kleinschreibung beim Vergleich keine Rolle spielen soll und Wildcards eingesetzt werden sollen, verwenden Sie die **wildmatch**-Funktion.

Beispiel: Ladeskript mit Verwendung von mixmatch

Beispiel: Ladeskript

Ladeskript

Sie können mixmatch zum Laden einer Teilmenge von Daten verwenden. Sie können zum Beispiel einen numerischen Wert für eine Formel in der Funktion liefern. Dann können Sie die geladenen Daten basierend auf dem numerischen Wert beschränken. Mixmatch liefert 0, wenn keine Übereinstimmung vorliegt. Alle Formeln, für die in diesem Beispiel keine Übereinstimmung vorliegt, liefern daher 0 und werden aus dem Datenladen mit dem WHERE-Befehl ausgeschlossen.

Erstellen Sie eine neue Registerkarte im Dateneditor und laden Sie die folgenden Daten als Inline-Ladevorgang: Erstellen Sie die Tabelle unten in Qlik Sense, um die Ergebnisse anzuzeigen.

```
Load * Inline [ transaction_id, transaction_date, transaction_amount, transaction_quantity,
customer_id, size, color_code 3750, 20180830, 23.56, 2, 2038593, L, Red 3751, 20180907,
556.31, 6, 203521, m, orange 3752, 20180916, 5.75, 1, 5646471, S, blue 3753, 20180922, 125.00,
7, 3036491, l, Black 3754, 20180922, 484.21, 13, 049681, xs, Red 3756, 20180922, 59.18, 2,
2038593, M, Blue 3757, 20180923, 177.42, 21, 203521, XL, Black ];  /* Create new table called
Transaction_Buckets Create new fields called Customer, and Color code - Black, Blue, blue Load
Transactions table. Mixmatch returns 1 for 'Black', 2 for 'Blue'.  Also returns 3 for 'blue'
because mixmatch is not case sensitive. Only values that returned numeric value greater than 0
 are loaded by WHERE statement into Transactions_Buckets table. */  Transaction_Buckets: Load
  customer_id,     customer_id as [Customer],    color_code as [Color Code - Black, Blue,
blue] Resident Transactions Where mixmatch(color_code,'Black','Blue') > 0;
```

Ergebnisse

Qlik Sense table showing the output from using the mixmatch function in the load script.

Color Code Black, Blue, blue	Customer
Black	203521
Black	3036491
Blue	2038593
blue	5646471

Beispiele: Diagrammformeln mit Verwendung von mixmatch

Beispiele: Diagrammformeln

Erstellen Sie eine neue Registerkarte im Dateneditor und laden Sie die folgenden Daten als Inline-Ladevorgang: Erstellen Sie nach dem Laden der Daten die Diagrammausdruckbeispiele unten in einer Qlik Sense-Tabelle.

Diagrammformel 1

```
MyTable: Load * inline [Cities, Count Toronto, 123 Toronto, 234 Toronto, 231 Boston, 32
Boston, 23 Boston, 1341 Beijing, 234 Beijing, 45 Beijing, 235 Stockholm, 938 Stockholm, 39
Stockholm, 189 zurich, 2342 zurich, 9033 zurich, 0039];
```

Die erste Formel in der Tabelle unten liefert 0 für „Stockholm“, da „Stockholm“ nicht in der Liste der Formeln in der **mixmatch**-Funktion enthalten ist. Es wird 4 für „Zurich“ geliefert, weil der **mixmatch**-Vergleich die Groß- und Kleinschreibung nicht berücksichtigt.

Qlik Sense Tabelle mit Beispielen der *mixmatch*-Funktion in einer Diagrammformel

Cities	**mixmatch(Cities,'Toronto','Boston','Beijing','Zurich')**	**mixmatch(Cities,'Toronto','Boston','Beijing','Stockholm','Zurich')**
Beijing	3	3
Boston	2	2
Stockholm	0	4
Toronto	1	1
zurich	4	5

Diagrammformel 2

Sie können mixmatch verwenden, um eine benutzerdefinierte Sortierung für eine Formel auszuführen.

Standardmäßig werden Spalten abhängig von den Daten alphabetisch oder numerisch sortiert.

Qlik Sense Tabelle mit einem Beispiel für die Standardsortierrreihenfolge

Cities
Beijing
Boston
Stockholm
Toronto
zurich

Gehen Sie wie folgt vor, um die Reihenfolge zu ändern:

1. Öffnen Sie den Abschnitt **Sortieren** für Ihr Diagramm im Fenster **Eigenschaften**.
2. Deaktivieren Sie die automatische Sortierung für die Spalte, in der Sie eine benutzerdefinierte Sortierung durchführen möchten.
3. Heben Sie die Auswahl von **Numerisch sortieren** und **Alphabetisch sortieren** auf.
4. Wählen Sie **Nach Formel sortieren** und geben Sie dann die folgende Formel ein:
   `=mixmatch( Cities, 'Toronto','Boston','Beijing','Stockholm','Zurich')`
   Die Sortierreihenfolge der Spalte Cities ändert sich.

Qlik Sense Tabelle mit einem Beispiel für das Ändern der Sortierreihenfolge mit der *mixmatch*-Funktion.

Cities
Toronto
Boston
Beijing

Cities
Stockholm
zurich

Sie können auch den gelieferten numerischen Wert anzeigen.

Qlik Sense Tabelle mit einem Beispiel für numerische Werte, die von der *mixmatch*-Funktion geliefert werden.

Cities	Cities & ' - ' & mixmatch (Cities, 'Toronto','Boston', 'Beijing','Stockholm','Zurich')
Toronto	Toronto - 1
Boston	Boston - 2
Beijing	Beijing - 3
Stockholm	Stockholm - 4
zurich	zurich - 5

pick

Die pick-Funktion liefert die *n*-te Formel in der Liste.

Syntax:

```
pick(n, expr1[ , expr2,...exprN])
```

Argumente:

Argumente

Argument	Beschreibung
n	*n* ist eine ganze Zahl zwischen 1 und N.

Beispiel:

Beispiel

Beispiel	Ergebnis
`pick( N, 'A','B',4, 6 )`	liefert 'B', wenn N = 2 liefert 4, wenn N = 3

wildmatch

Die **wildmatch**-Funktion vergleicht den ersten Parameter mit allen folgenden und liefert die übereinstimmenden Formeln. Sie lässt die Verwendung von Platzhalterzeichen (***** und **?**) in den Vergleichsstrings zu. ***** entspricht jeder Abfolge von Zeichen. **?** entspricht einem einzelnen Zeichen. Beim Vergleich wird die Groß- und Kleinschreibung berücksichtigt.

Syntax:

```
wildmatch( str, expr1 [ , expr2,...exprN ])
```

Wenn der Vergleich ohne Wildcards durchgeführt werden soll, verwenden Sie die **match**- oder **mixmatch**-Funktion.

Beispiel: Ladeskript mit Verwendung von wildmatch

Beispiel: Ladeskript

Ladeskript

Sie können wildmatch zum Laden einer Teilmenge von Daten verwenden. Sie können zum Beispiel einen numerischen Wert für eine Formel in der Funktion liefern. Dann können Sie die geladenen Daten basierend auf dem numerischen Wert beschränken. Wildmatch liefert 0, wenn keine Übereinstimmung vorliegt. Alle Formeln, für die in diesem Beispiel keine Übereinstimmung vorliegt, liefern daher 0 und werden aus dem Datenladen mit dem WHERE-Befehl ausgeschlossen.

Erstellen Sie eine neue Registerkarte im Dateneditor und laden Sie die folgenden Daten als Inline-Ladevorgang: Erstellen Sie die Tabelle unten in Qlik Sense, um die Ergebnisse anzuzeigen.

```
Transactions: Load * Inline [ transaction_id, transaction_date, transaction_amount,
transaction_quantity, customer_id, size, color_code 3750, 20180830, 23.56, 2, 2038593, L, Red
3751, 20180907, 556.31, 6, 203521, m, orange 3752, 20180916, 5.75, 1, 5646471, S, blue 3753,
20180922, 125.00, 7, 3036491, l, Black 3754, 20180922, 484.21, 13, 049681, xs, Red 3756,
20180922, 59.18, 2, 2038593, M, Blue 3757, 20180923, 177.42, 21, 203521, XL, Black ];   /*
Create new table called Transaction_Buckets Create new fields called Customer, and Color code
- Black, Blue, blue, red Load Transactions table. Wildmatch returns 1 for 'Black', 'Blue', and
'blue', and 2 for 'Red'.  Only values that returned numeric value greater than 0  are loaded
by WHERE statement into Transactions_Buckets table. */   Transaction_Buckets: Load
customer_id,    customer_id as [Customer],    color_code as [Color Code Black, Blue, blue,
Red] Resident Transactions Where wildmatch(color_code,'Bl*','R??') > 0;
```

Ergebnisse

Qlik SenseTabelle mit der Ausgabe der *wildmatch*-Funktion im Ladeskript

Color Code Black, Blue, blue, Red	Customer
Black	203521
Black	3036491
Blue	2038593
blue	5646471
Red	049681
Red	2038593

Beispiele: Diagrammformeln mit Verwendung von wildmatch

Beispiel: Diagrammformel

Diagrammformel 1

Erstellen Sie eine neue Registerkarte im Dateneditor und laden Sie die folgenden Daten als Inline-Ladevorgang: Erstellen Sie nach dem Laden der Daten die Diagrammausdruckbeispiele unten in einer Qlik Sense-Tabelle.

```
MyTable: Load * inline [Cities, Count Toronto, 123 Toronto, 234 Toronto, 231 Boston, 32 Boston, 23 Boston, 1341 Beijing, 234 Beijing, 45 Beijing, 235 Stockholm, 938 Stockholm, 39 Stockholm, 189 zurich, 2342 zurich, 9033 zurich, 0039];
```

Die erste Formel in der Tabelle unten liefert 0 für „Stockholm“, da „Stockholm“ nicht in der Liste der Formeln in der **wildmatch**-Funktion enthalten ist. Sie gibt auch 0 für „Boston“ zurück, weil ? nur für ein einziges Zeichen übereinstimmt.

Qlik Sense Tabelle mit Beispielen der *wildmatch*-Funktion in einer Diagrammformel

Cities	**wildmatch(Cities,'Tor*','?ton','Beijing','*urich')**	**wildmatch(Cities,'Tor*','???ton','Beijing','Stockholm','*urich')**
Beijing	3	3
Boston	0	2
Stockholm	0	4
Toronto	1	1
zurich	4	5

Diagrammformel 2

Sie können wildmatch verwenden, um eine benutzerdefinierte Sortierung für eine Formel auszuführen.

Standardmäßig werden Spalten abhängig von den Daten numerisch oder alphabetisch sortiert.

Qlik Sense Tabelle mit einem Beispiel für die Standardsortierrreihenfolge

Cities
Beijing
Boston
Stockholm
Toronto
zurich

Gehen Sie wie folgt vor, um die Reihenfolge zu ändern:

1. Öffnen Sie den Abschnitt **Sortieren** für Ihr Diagramm im Fenster **Eigenschaften**.
2. Deaktivieren Sie die automatische Sortierung für die Spalte, in der Sie eine benutzerdefinierte Sortierung durchführen möchten.
3. Heben Sie die Auswahl von **Numerisch sortieren** und **Alphabetisch sortieren** auf.
4. Wählen Sie **Nach Formel sortieren** und geben Sie dann eine Formel wie die Folgende ein:
   `=wildmatch( Cities, 'Tor*','???ton','Beijing','Stockholm','*urich')`
   Die Sortierreihenfolge der Spalte Cities ändert sich.

Qlik Sense Tabelle mit einem Beispiel für das Ändern der Sortierreihenfolge mit der *wildmatch*-Funktion.

Cities
Toronto
Boston
Beijing
Stockholm
zurich

Sie können auch den gelieferten numerischen Wert anzeigen.

Qlik Sense Tabelle mit einem Beispiel für numerische Werte, die von der *wildmatch*-Funktion geliefert werden

Cities	**Cities & ' - ' & wildmatch (Cities, 'Tor*','???ton','Beijing','Stockholm','*urich')**
Toronto	Toronto - 1
Boston	Boston - 2
Beijing	Beijing - 3
Stockholm	Stockholm - 4
zurich	zurich - 5

8.6 Counter-Funktionen

In diesem Abschnitt werden Funktionen in Hinblick auf Datensatz-Häufigkeiten während der Evaluierung von **LOAD**-Befehlen im Datenladeskript beschrieben. Die einzige Funktion, die in Diagrammformeln verwendet werden kann, ist **RowNo()**.

Einige Counter-Funktionen haben keine Parameter, die nachstehenden Klammern sind dennoch erforderlich.

Counter-Funktionen – Übersicht

Jede Funktion wird nach der Übersicht genauer beschrieben. Sie können auch auf den Funktionsnamen in der Syntax klicken, um direkt auf die Details zu der spezifischen Funktion zuzugreifen.

autonumber
Diese Skriptfunktion liefert eine eindeutige ganze Zahl für jeden distinkten ausgewerteten Wert von *expression* bei der Skriptausführung. Diese Funktion kann beispielsweise verwendet werden, um zusammengesetzte Schlüssel zu vereinfachen oder abzukürzen.

```
autonumber (expression[ , AutoID])
```

autonumberhash128
Diese Skriptfunktion berechnet für jede Kombination von Formeln einen 128-Bit-Hash-Wert und liefert für jeden unterschiedlichen Hash-Wert innerhalb des Skripts eine eindeutige ganze Zahl. Diese Funktion kann beispielsweise verwendet werden, um zusammengesetzte Schlüssel zu vereinfachen oder abzukürzen.

```
autonumberhash128 (expression {, expression})
```

autonumberhash256
Diese Skriptfunktion berechnet für jede Kombination von Formeln einen 256-Bit-Hash-Wert und liefert für jeden unterschiedlichen Hash-Wert innerhalb des Skripts eine eindeutige ganze Zahl. Diese Funktion kann beispielsweise verwendet werden, um zusammengesetzte Schlüssel zu vereinfachen oder abzukürzen.

```
autonumberhash256 (expression {, expression})
```

IterNo
Diese Skriptfunktion liefert eine ganze Zahl, die angibt, wie oft ein Datensatz durch einen **LOAD**-Befehl mit **while**-Zusatz geladen wird. Das erste Einlesen zählt als Nummer 1. Die Funktion **IterNo** ist nur in Verbindung mit einer **while**-Bedingung nützlich.

```
IterNo ( )
```

RecNo
Diese Skriptfunktion liefert eine ganze Zahl, die die Nummer der gerade gelesenen Zeile in der aktuellen Tabelle repräsentiert. Der erste Datensatz trägt die Nummer 1.

```
RecNo ( )
```

RowNo - script function
Diese Funktion gibt eine ganze Zahl zurück, welche die Position der aktuellen Zeile in der entstehenden internen Qlik Sense-Tabelle angibt. Die erste Zeile trägt die Nummer 1.

```
RowNo ( )
```

RowNo - chart function
RowNo() liefert die Anzahl der aktuellen Zeilen im aktuellen Spaltenabschnitt in einer Tabelle. In Bitmap-Diagrammen liefert **RowNo()** die Zahl der aktuellen Zeile im entsprechenden Tabellendiagramm des Diagramms.

```
RowNo - Diagrammfunktion([TOTAL])
```

autonumber

Diese Skriptfunktion liefert eine eindeutige ganze Zahl für jeden distinkten ausgewerteten Wert von *expression* bei der Skriptausführung. Diese Funktion kann beispielsweise verwendet werden, um zusammengesetzte Schlüssel zu vereinfachen oder abzukürzen.

*Sie können nur die **autonumber**-Schlüssel verbinden, die bei demselben Datenladevorgang generiert wurden, da die Ganzzahl in der Reihenfolge generiert wird, in der die Tabelle eingelesen wird. Sie müssen Schlüssel verwenden, die unabhängig von der Quelldatensortierung zwischen Datenladevorgängen beibehalten werden. Dazu eignen sich die Funktionen **hash128**, **hash160** oder **hash256**.*

Syntax:

```
autonumber(expression[ , AutoID])
```

Argumente:

Argument	Beschreibung
AutoID	Um mehrere Zählerinstanzen zu erstellen, wenn die **autonumber**-Funktion im Skript bei verschiedenen Schlüsseln verwendet wird, kann der optionale Parameter *AutoID* zum Benennen der einzelnen Zähler genutzt werden.

Beispiel: Erstellen eines zusammengesetzten Schlüssels

In diesem Beispiel erstellen wir einen zusammengesetzten Schlüssel mithilfe der Funktion **autonumber**, damit weniger Speicherplatz erforderlich ist. Das Beispiel ist für Demonstrationszwecke nur kurz, wäre aber besonders in einer Tabelle mit vielen Zeilen sinnvoll.

Beispieldaten

Region	Year	Month	Sales
North	2014	May	245
North	2014	May	347
North	2014	June	127
South	2014	June	645
South	2013	May	367
South	2013	May	221

Die Quelldaten werden mithilfe von Inline-Daten geladen. Anschließend wird der vorangehende Load-Befehl hinzugefügt, wodurch ein zusammengesetzter Schlüssel aus den Feldern Region, Year und Month erstellt wird.

```
RegionSales:
LOAD *,
AutoNumber(Region&Year&Month) as RYMkey;

LOAD * INLINE
[ Region, Year, Month, Sales
North, 2014,   May,    245
North, 2014,   May,    347
North, 2014,   June,   127
South, 2014,   June,   645
```

```
South, 2013,   May, 367
South, 2013,   May,    221
];
```

Die sich ergebende Tabelle sieht folgendermaßen aus:

Ergebnistabelle

Region	Year	Month	Sales	RYMkey
North	2014	May	245	1
North	2014	May	347	1
North	2014	June	127	2
South	2014	June	645	3
South	2013	May	367	4
South	2013	May	221	4

In diesem Beispiel können Sie auf den RYM-Schlüssel, zum Beispiel 1, anstelle des Strings 'North2014May' Bezug nehmen, wenn Sie eine Verknüpfung zu einer weiteren Tabelle erstellen möchten.

Jetzt wird eine Quelltabelle mit Kosten auf ähnliche Weise geladen. Die Felder Region, Year und Month werden im vorhergehenden Ladevorgang ausgeschlossen, damit kein synthetischer Schlüssel erstellt wird; es wird dabei bereits ein zusammengesetzter Schlüssel mit der Funktion **autonumber** erstellt, was die Tabelle verlinkt.

```
RegionCosts:
LOAD Costs,
AutoNumber(Region&Year&Month) as RYMkey;

LOAD * INLINE
[ Region, Year, Month, Costs
South, 2013,   May,    167
North, 2014,   May,    56
North, 2014,   June,   199
South, 2014,   June,   64
South, 2013,   May, 172
South, 2013,   May,    126
];
```

Jetzt können zum Arbeitsblatt eine Tabellenvisualisierung und die Felder Region, Year und Month sowie Summenkennzahlen für Umsatz und Kosten hinzugefügt werden. Die Tabelle sieht folgendermaßen aus:

Ergebnistabelle

Region	Year	Month	Sum([Sales])	Sum([Costs])
Totals	-	-	1952	784
North	2014	June	127	199

Region	Year	Month	Sum([Sales])	Sum([Costs])
North	2014	May	592	56
South	2014	June	645	64
South	2013	May	588	465

autonumberhash128

Diese Skriptfunktion berechnet für jede Kombination von Formeln einen 128-Bit-Hash-Wert und liefert für jeden unterschiedlichen Hash-Wert innerhalb des Skripts eine eindeutige ganze Zahl. Diese Funktion kann beispielsweise verwendet werden, um zusammengesetzte Schlüssel zu vereinfachen oder abzukürzen.

*Sie können nur die **autonumberhash128**-Schlüssel verbinden, die bei demselben Datenladevorgang generiert wurden, da die Ganzzahl in der Reihenfolge generiert wird, in der die Tabelle eingelesen wird. Sie müssen Schlüssel verwenden, die unabhängig von der Quelldatensortierung zwischen Datenladevorgängen beibehalten werden. Dazu eignen sich die Funktionen **hash128**, **hash160** oder **hash256**.*

Syntax:

```
autonumberhash128(expression {, expression})
```

Beispiel: Erstellen eines zusammengesetzten Schlüssels

In diesem Beispiel erstellen wir einen zusammengesetzten Schlüssel mithilfe der Funktion **autonumberhash128**, damit weniger Speicherplatz erforderlich ist. Das Beispiel ist für Demonstrationszwecke nur kurz, wäre aber besonders in einer Tabelle mit vielen Zeilen sinnvoll.

Beispieldaten

Region	Year	Month	Sales
North	2014	May	245
North	2014	May	347
North	2014	June	127
South	2014	June	645
South	2013	May	367
South	2013	May	221

Die Quelldaten werden mithilfe von Inline-Daten geladen. Anschließend wird der vorangehende Load-Befehl hinzugefügt, wodurch ein zusammengesetzter Schlüssel aus den Feldern Region, Year und Month erstellt wird.

```
RegionSales:
LOAD *,
AutoNumberHash128(Region, Year, Month) as RYMkey;
```

```
LOAD * INLINE
[ Region, Year, Month, Sales
North, 2014,   May,    245
North, 2014,   May,    347
North, 2014,   June,   127
South, 2014,   June,   645
South, 2013,   May, 367
South, 2013,   May,    221
];
```

Die sich ergebende Tabelle sieht folgendermaßen aus:

Ergebnistabelle

Region	Year	Month	Sales	RYMkey
North	2014	May	245	1
North	2014	May	347	1
North	2014	June	127	2
South	2014	June	645	3
South	2013	May	367	4
South	2013	May	221	4

In diesem Beispiel können Sie auf den RYM-Schlüssel, zum Beispiel 1, anstelle des Strings 'North2014May' Bezug nehmen, wenn Sie eine Verknüpfung zu einer weiteren Tabelle erstellen möchten.

Jetzt wird eine Quelltabelle mit Kosten auf ähnliche Weise geladen. Die Felder Region, Year und Month werden im vorhergehenden Ladevorgang ausgeschlossen, damit kein synthetischer Schlüssel erstellt wird; es wird dabei bereits ein zusammengesetzter Schlüssel mit der Funktion **autonumberhash128** erstellt, was die Tabelle verlinkt.

```
RegionCosts:
LOAD Costs,
AutoNumberHash128(Region, Year, Month) as RYMkey;

LOAD * INLINE
[ Region, Year, Month, Costs
South, 2013,   May,    167
North, 2014,   May,    56
North, 2014,   June,   199
South, 2014,   June,   64
South, 2013,   May, 172
South, 2013,   May,    126
];
```

Jetzt können zum Arbeitsblatt eine Tabellenvisualisierung und die Felder Region, Year und Month sowie Summenkennzahlen für Umsatz und Kosten hinzugefügt werden. Die Tabelle sieht folgendermaßen aus:

Ergebnistabelle

Region	Year	Month	Sum([Sales])	Sum([Costs])
Totals	-	-	1952	784
North	2014	June	127	199
North	2014	May	592	56
South	2014	June	645	64
South	2013	May	588	465

autonumberhash256

Diese Skriptfunktion berechnet für jede Kombination von Formeln einen 256-Bit-Hash-Wert und liefert für jeden unterschiedlichen Hash-Wert innerhalb des Skripts eine eindeutige ganze Zahl. Diese Funktion kann beispielsweise verwendet werden, um zusammengesetzte Schlüssel zu vereinfachen oder abzukürzen.

*Sie können nur die **autonumberhash256**-Schlüssel verbinden, die bei demselben Datenladevorgang generiert wurden, da die Ganzzahl in der Reihenfolge generiert wird, in der die Tabelle eingelesen wird. Sie müssen Schlüssel verwenden, die unabhängig von der Quelldatensortierung zwischen Datenladevorgängen beibehalten werden. Dazu eignen sich die Funktionen **hash128**, **hash160** oder **hash256**.*

Syntax:

```
autonumberhash256(expression {, expression})
```

Beispiel: Erstellen eines zusammengesetzten Schlüssels

In diesem Beispiel erstellen wir einen zusammengesetzten Schlüssel mithilfe der Funktion **autonumberhash256**, damit weniger Speicherplatz erforderlich ist. Das Beispiel ist für Demonstrationszwecke nur kurz, wäre aber besonders in einer Tabelle mit vielen Zeilen sinnvoll.

Beispieltabelle

Region	Year	Month	Sales
North	2014	May	245
North	2014	May	347
North	2014	June	127
South	2014	June	645
South	2013	May	367
South	2013	May	221

Die Quelldaten werden mithilfe von Inline-Daten geladen. Anschließend wird der vorangehende Load-Befehl hinzugefügt, wodurch ein zusammengesetzter Schlüssel aus den Feldern Region, Year und Month erstellt wird.

```
RegionSales:
LOAD *,
AutoNumberHash256(Region, Year, Month) as RYMkey;

LOAD * INLINE
[ Region, Year, Month, Sales
North, 2014,   May,    245
North, 2014,   May,    347
North, 2014,   June,   127
South, 2014,   June,   645
South, 2013,   May, 367
South, 2013,   May,    221
];
```

Die sich ergebende Tabelle sieht folgendermaßen aus:

Ergebnistabelle

Region	Year	Month	Sales	RYMkey
North	2014	May	245	1
North	2014	May	347	1
North	2014	June	127	2
South	2014	June	645	3
South	2013	May	367	4
South	2013	May	221	4

In diesem Beispiel können Sie auf den RYM-Schlüssel, zum Beispiel 1, anstelle des Strings 'North2014May' Bezug nehmen, wenn Sie eine Verknüpfung zu einer weiteren Tabelle erstellen möchten.

Jetzt wird eine Quelltabelle mit Kosten auf ähnliche Weise geladen. Die Felder Region, Year und Month werden im vorhergehenden Ladevorgang ausgeschlossen, damit kein synthetischer Schlüssel erstellt wird; es wird dabei bereits ein zusammengesetzter Schlüssel mit der Funktion **autonumberhash256** erstellt, was die Tabelle verlinkt.

```
RegionCosts:
LOAD Costs,
AutoNumberHash256(Region, Year, Month) as RYMkey;

LOAD * INLINE
[ Region, Year, Month, Costs
South, 2013,   May,    167
North, 2014,   May,    56
North, 2014,   June,   199
South, 2014,   June,   64
South, 2013,   May, 172
South, 2013,   May,    126
];
```

Jetzt können zum Arbeitsblatt eine Tabellenvisualisierung und die Felder Region, Year und Month sowie Summenkennzahlen für Umsatz und Kosten hinzugefügt werden. Die Tabelle sieht folgendermaßen aus:

Ergebnistabelle

Region	Year	Month	Sum([Sales])	Sum([Costs])
Totals	-	-	1952	784
North	2014	June	127	199
North	2014	May	592	56
South	2014	June	645	64
South	2013	May	588	465

IterNo

Diese Skriptfunktion liefert eine ganze Zahl, die angibt, wie oft ein Datensatz durch einen **LOAD**-Befehl mit **while**-Zusatz geladen wird. Das erste Einlesen zählt als Nummer 1. Die Funktion **IterNo** ist nur in Verbindung mit einer **while**-Bedingung nützlich.

Syntax:

```
IterNo( )
```

Beispiele und Ergebnisse:

Beispiel:

```
LOAD
   IterNo() as Day,
   Date( StartDate + IterNo() - 1 ) as Date
   While StartDate + IterNo() - 1 <= EndDate;

LOAD * INLINE
[StartDate, EndDate
2014-01-22, 2014-01-26
];
```

Der **LOAD**-Befehl generiert einen Datensatz pro Datum innerhalb des von **StartDate** und **EndDate** definierten Bereichs.

In diesem Beispiel sieht die sich ergebende Tabelle folgendermaßen aus:

Ergebnistabelle

Day	Date
1	2014-01-22
2	2014-01-23
3	2014-01-24
4	2014-01-25
5	2014-01-26

RecNo

Diese Skriptfunktion liefert eine ganze Zahl, die die Nummer der gerade gelesenen Zeile in der aktuellen Tabelle repräsentiert. Der erste Datensatz trägt die Nummer 1.

Syntax:

```
RecNo( )
```

Im Gegensatz zu **RowNo( )**, wobei die Zeilen in der sich ergebenden Qlik Sense-Tabelle gezählt werden, zählt **RecNo( )** die Datensätze in der Rohdatentabelle und wird zurückgesetzt, wenn eine Rohdatentabelle mit einer anderen zusammengefasst wird.

Beispiel: Datenladeskript

Ladevorgang bei Rohdatentabellen:

```
Tab1:
LOAD * INLINE
[A, B
1, aa
2,cc
3,ee];

Tab2:
LOAD * INLINE
[C, D
5, xx
4,yy
6,zz];
```

Laden von Datensatz und Zeilennummer bei ausgewählten Zeilen:

```
QTab:
LOAD *,
RecNo( ),
RowNo( )
resident Tab1 where A<>2;

LOAD
C as A,
D as B,
```

```
RecNo( ),
RowNo( )
resident Tab2 where A<>5;

//We don't need the source tables anymore, so we drop them
Drop tables Tab1, Tab2;
```

Die resultierende interne Qlik Sense-Tabelle:

Ergebnistabelle

A	B	RecNo( )	RowNo( )
1	aa	1	1
3	ee	3	2
4	yy	2	3
6	zz	3	4

RowNo

Diese Funktion gibt eine ganze Zahl zurück, welche die Position der aktuellen Zeile in der entstehenden internen Qlik Sense-Tabelle angibt. Die erste Zeile trägt die Nummer 1.

Syntax:

```
RowNo( [TOTAL])
```

Im Gegensatz zu **RecNo( )**, welche die Datensätze in der Rohdatentabelle zählt, zählt die Funktion **RowNo( )** die Datensätze nicht mit, die durch **where**-Bedingungen ausgeschlossen sind. Sie wird nicht zurückgesetzt, wenn eine Rohdatentabelle mit einer anderen zusammengefasst wird.

> *Wenn Sie einen vorangehenden Load-Befehl verwenden, also mehrere **LOAD** -Befehle, die aus derselben Tabelle lesen, gestapelt ausgeführt werden, können Sie **RowNo( )** nur im obersten **LOAD** -Befehl verwenden. Wenn Sie **RowNo( )** in nachfolgenden **LOAD** -Befehlen verwenden, wird 0 zurückgegeben.*

Beispiel: Datenladeskript

Ladevorgang bei Rohdatentabellen:

```
Tab1:
LOAD * INLINE
[A, B
1, aa
2,cc
3,ee];

Tab2:
LOAD * INLINE
[C, D
5, xx
```

```
4,yy
6,zz];
```

Laden von Datensatz und Zeilennummer bei ausgewählten Zeilen:

```
QTab:

LOAD *,

RecNo( ),

RowNo( )

resident Tab1 where A<>2;



LOAD

C as A,

D as B,

RecNo( ),

RowNo( )

resident Tab2 where A<>5;



//We don't need the source tables anymore, so we drop them

Drop tables Tab1, Tab2;
```

Die resultierende interne Qlik Sense-Tabelle:

Ergebnistabelle

A	B	RecNo( )	RowNo( )
1	aa	1	1
3	ee	3	2
4	yy	2	3
6	zz	3	4

RowNo - Diagrammfunktion

RowNo() liefert die Anzahl der aktuellen Zeilen im aktuellen Spaltenabschnitt in einer Tabelle. In Bitmap-Diagrammen liefert **RowNo()** die Zahl der aktuellen Zeile im entsprechenden Tabellendiagramm des Diagramms.

Hat das Diagramm dagegen mehrere vertikale Dimensionen, so umfasst der Spaltenabschnitt nur Zeilen, deren Werte in allen Dimensionen mit Ausnahme der in der Priorität der Sortierfolgen letzten Dimension übereinstimmen.

Spaltensegmente

Region	Country	Population	Rank(Population)
Americas	Mexico	128,932,753	2
Americas	Canada	37,742,154	3
Americas	United States of America	331,002,651	1
Europe	Sweden	10,099,265	4
Europe	United Kingdom	67,886,011	2
Europe	France	65,273,511	3
Europe	Germany	83,783,942	1

Das Sortieren nach y-Werten in Diagrammen oder nach Formelspalten in Tabellen ist nicht zulässig, wenn diese Diagrammfunktion in einer der Diagrammformeln verwendet wird. Diese Sortierungsoptionen werden daher automatisch deaktiviert. Wenn Sie diese Diagrammfunktion in einer Visualisierung oder Tabelle verwenden, wird die Sortierung der Visualisierung auf die sortierte Eingabe dieser Funktion zurückgesetzt.

Syntax:

```
RowNo([TOTAL])
```

Rückgabe Datentyp: ganze Zahl

Argumente:

Argument	Beschreibung
TOTAL	Wenn das Diagramm nur eine Dimension hat oder die Formel mit dem Zusatz **TOTAL** als Argument versehen ist, entspricht der Spaltenabschnitt der gesamten Spalte.

Beispiel: Diagrammformel mit RowNo

Beispiel – Diagrammformel

Ladeskript

Laden Sie die folgenden Daten als Inline-Ladevorgang in den Dateneditor, um die folgenden Diagrammformelbeispiele zu erstellen.

```
Temp:
LOAD * inline [
Customer|Product|OrderNumber|UnitSales|UnitPrice
Astrida|AA|1|4|16
Astrida|AA|7|10|15
Astrida|BB|4|9|9
Betacab|CC|6|5|10
Betacab|AA|5|2|20
Betacab|BB|1|25| 25
Canutility|AA|3|8|15
Canutility|CC|5|4|19
Divadip|CC|2|4|16
```

```
Divadip|DD|3|1|25
] (delimiter is '|');
```

Diagrammformel

Erstellen Sie eine Tabellenvisualisierung in einem Qlik SenseArbeitsblatt mit **Customer** und **UnitSales** als Dimensionen. Fügen Sie `RowNo( )` und `RowNo(TOTAL)` als Kennzahlen mit den Beschriftungen Row in Segment bzw. **Row Number** hinzu. Fügen Sie die folgende Formel als Kennzahl zur Tabelle hinzu.

```
If( RowNo( )=1, 0, UnitSales / Above( UnitSales ))
```

Ergebnis

Customer	UnitSales	Row in Segment	Row Number	If(RowNo( )=1, 0, UnitSales / Above(UnitSales))
Astrida	4	1	1	0
Astrida	9	2	2	2.25
Astrida	10	3	3	1.1111111111111
Betacab	2	1	4	0
Betacab	5	2	5	2.5
Betacab	25	3	6	5
Canutility	4	1	7	0
Canutility	8	2	8	2
Divadip	1	1	9	0
Divadip	4	2	10	4

Erläuterung

Die Spalte **Row in Segment** zeigt die Ergebnisse 1, 2, 3 für den Spaltenabschnitt an, der die Werte von UnitSales für den Kunden Astrida enthält. Die Zeilennummerierung beginnt dann für den nächsten Spaltenabschnitt, Betacab, wieder bei 1.

Die Spalte **Row Number** berücksichtigt die Dimensionen aufgrund des Arguments `TOTAL` für `RowNo()` nicht und zählt die Zeilen in der Tabelle.

Diese Formel liefert 0 für die erste Zeile in jedem Spaltenabschnitt, also zeigt die Spalte Folgendes an:

0, 2.25, 1.1111111, 0, 2.5, 5, 0, 2, 0 und 4.

Siehe auch:

Above - Diagrammfunktion (page 1303)

8.7 Funktionen für Datum und Uhrzeit

Qlik Sense-Funktionen für Datum und Zeit werden für die Umwandlung und Konvertierung von Daten- und Zeitwerten verwendet. Alle Funktionen können sowohl im Datenladeskript als auch in den Diagrammformeln verwendet werden.

Die Funktionen basieren auf einer laufenden Nummer von Datum und Zeit, die der Zahl an Tagen seit dem 30. Dezember 1899 entspricht. Die ganze Zahl steht für den Tag und der Bruchteil für die Uhrzeit des Tages.

Qlik Sense verwendet den numerischen Wert des Parameters. Also gilt eine Zahl auch als Parameter, wenn sie nicht als Datum oder Uhrzeit formatiert ist. Entspricht der Parameter nicht dem numerischen Wert, zum Beispiel, weil es sich um einen String handelt, versucht Qlik Sense, den String gemäß der Umgebungsvariablen für Datum und Uhrzeit zu interpretieren.

Wenn das verwendete Uhrzeitformat im Parameter nicht mit dem Uhrzeitformat der Umgebungsvariablen übereinstimmt, kann Qlik Sense die Uhrzeit nicht korrekt interpretieren. In diesem Fall können Sie entweder die Systemeinstellungen ändern oder Sie verwenden die Interpretationsfunktion.

In den Beispielen für jede Funktion werden die Standardzeit- und -datumsformate hh:mm:ss und YYYY-MM-DD (ISO 8601) angenommen.

Wenn ein Zeitstempel mit einer Datums- oder Zeitfunktion verarbeitet wird, ignoriert Qlik Sense etwaige Parameter für die Sommerzeit, es sei denn, die Datums- oder Zeitfunktion enthält eine geografische Position.

Beispiel: `ConvertToLocalTime( filetime('Time.qvd'), 'Paris')` verwendet Parameter für die Sommerzeit, während `ConvertToLocalTime(filetime('Time.qvd'), 'GMT-01:00')` keine Parameter für die Sommerzeit verwendet.

Funktionen für Datum und Uhrzeit – Übersicht

Jede Funktion wird nach der Übersicht genauer beschrieben. Sie können auch auf den Funktionsnamen in der Syntax klicken, um direkt auf die Details zu der spezifischen Funktion zuzugreifen.

Ganzzahlausdrücke für die Uhrzeit

second
Diese Funktion liefert die Sekunden als ganze Zahl, wenn **expression** entsprechend dem Standardformat als Uhrzeit interpretiert wird.

```
second (expression)
```

minute
Diese Funktion liefert die Minute als ganze Zahl, wenn **expression** entsprechend dem Standardformat als Uhrzeit interpretiert wird.

```
minute (expression)
```

hour
Diese Funktion liefert die Stunde als ganze Zahl, wenn **expression** entsprechend dem Standardformat als Uhrzeit interpretiert wird.

```
hour (expression)
```

day
Diese Funktion liefert den Tag als ganze Zahl, wenn **expression** entsprechend dem Standardformat als Datum interpretiert wird.

```
day (expression)
```

week
Diese Funktion liefert die Kalenderwoche als Ganzzahl gemäß ISO 8601. Die Kalenderwoche berechnet sich durch die Datumsinterpretation der Formel entsprechend dem Standardformat.

```
week (expression)
```

month
Diese Funktion gibt einen dualen Wert zurück: ein Monatsname gemäß Definition in der Umgebungvariable **MonthNames** sowie eine Ganzzahl zwischen 1-12. Der Monat berechnet sich durch die Datumsinterpretation der Formel entsprechend dem Standardformat.

```
month (expression)
```

year
Diese Funktion liefert das Jahr als ganze Zahl, wenn **expression** entsprechend der Standardinterpretation als Datum interpretiert wird.

```
year (expression)
```

weekyear
Diese Funktion liefert das Jahr, zu dem die Wochennummer gemäß den Umgebungsvariablen zählt. Die Kalenderwochen bewegen sich zwischen 1 und circa 52.

```
weekyear (expression)
```

weekday
Diese Funktion liefert einen dualen Wert mit:

- Einem Wochentag wie in der Umgebungsvariable **DayNames** definiert.
- Einer ganzen Zahl zwischen 0-6 entsprechend den Tagen der Woche (0-6).

```
weekday (date)
```

Zeitstempelfunktionen

now
Diese Funktion gibt einen Zeitstempel der aktuellen Uhrzeit zurück. Die Funktion gibt Werte im Systemvariablenformat **TimeStamp** zurück. Der Standardwert von **timer_mode** ist 1.

```
now ([ timer_mode])
```

today
Diese Funktion gibt das aktuelle Datum zurück. Die Funktion gibt Werte im Systemvariablenformat `DateFormat` zurück.

```
today ([timer_mode])
```

LocalTime
Diese Funktion liefert einen Zeitstempel der aktuellen Uhrzeit, bezogen auf eine bestimmte Zeitzone.

```
localtime ([timezone [, ignoreDST ]])
```

Funktionen erstellen

makedate
Die Funktion liefert ein Datum bestehend aus der angegebenen Jahreszahl **YYYY**, dem Monat **MM** und dem Wochentag **DD**.

```
makedate (YYYY [ , MM [ , DD ] ])
```

makeweekdate
Diese Funktion gibt ein Datum zurück, das sich aus dem Jahr, der Wochennummer und dem Tag der Woche berechnet.

```
makeweekdate (YYYY [ , WW [ , D ] ])
```

maketime
Die Funktion liefert eine Zeit bestehend aus der angegebenen Stunde **hh**, der Minute **mm** und der Sekunde **ss**.

```
maketime (hh [ , mm [ , ss [ .fff ] ] ])
```

Sonstige Datumsfunktionen

AddMonths
Diese Funktion liefert das Datum, das **n** Monate nach **startdate** liegt, bzw. **n** Monate vor **startdate**, wenn **n** negativ ist.

```
addmonths (startdate, n , [ , mode])
```

AddYears
Diese Funktion liefert das Datum, das **n** Jahre nach **startdate** liegt, bzw. **n** Jahre vor **startdate**, wenn **n** negativ ist.

```
addyears (startdate, n)
```

yeartodate
Diese Funktion findet den Eingabe-Zeitstempel innerhalb des Jahres mit dem Datum, an welchem das Skript zuletzt aufgerufen wurde, und gibt True zurück, wenn der Zeitstempel gefunden wurde bzw. False, wenn er nicht gefunden wurde.

```
yeartodate (date [ , yearoffset [ , firstmonth [ , todaydate] ] ])
```

Zeitzonenfunktionen

timezone

Diese Funktion gibt die Zeitzone zurück, die für den Computer definiert ist, auf dem die Qlik-Engine ausgeführt wird.

```
timezone ( )
```

GMT

Diese Funktion liefert die aktuelle Greenwich Mean Time der Regionaleinstellungen.

```
GMT ( )
```

UTC

Liefert die aktuelle Coordinated Universal Time.

```
UTC ( )
```

daylightsaving

Liefert die aktuelle Einstellung bezüglich der Sommerzeit, so wie sie in Windows definiert ist.

```
daylightsaving ( )
```

converttolocaltime

Konvertiert einen UTC- oder GMT-Zeitstempel in eine lokale Zeit als dualen Wert. Der Standort kann eine beliebige Stadt und Ortsbezeichnung in jeder Zeitzone weltweit sein.

```
converttolocaltime (timestamp [, place [, ignore_dst=false]])
```

Funktionen zur Zeitfestlegung

setdateyear

Als Eingabe verwendet diese Funktion einen **timestamp** und ein **year** und aktualisiert den **timestamp** mit dem in der Eingabe festgelegten **year** .

```
setdateyear  (timestamp, year)
```

setdateyearmonth

Als Eingabe verwendet diese Funktion einen **timestamp**, einen **month** und ein **year** und aktualisiert den **timestamp** mit dem in der Eingabe festgelegten **year** und dem **month** .

```
setdateyearmonth  (timestamp, year, month)
```

In... Funktionen

inyear

Diese Funktion liefert True, wenn **timestamp** innerhalb des Jahres liegt, das **base_date** enthält.

```
inyear (date, basedate , shift [, first_month_of_year = 1])
```

inyeartodate
Diese Funktion liefert True, wenn **timestamp** in dem laufenden Jahr mit **base_date** bis einschließlich der letzten Millisekunde von **base_date** liegt.

```
inyeartodate (date, basedate , shift [, first_month_of_year = 1])
```

inquarter
Diese Funktion liefert True, wenn **timestamp** innerhalb des Quartals liegt, das **base_date** enthält.

```
inquarter (date, basedate , shift [, first_month_of_year = 1])
```

inquartertodate
Diese Funktion liefert True, wenn **timestamp** in dem Teil des Quartals liegt, der **base_date** enthält, und zwar bis einschließlich der letzten Millisekunde von **base_date**.

```
inquartertodate (date, basedate , shift [, first_month_of_year = 1])
```

inmonth
Diese Funktion liefert True, wenn **timestamp** innerhalb des Monats liegt, der **base_date** enthält.

```
inmonth (date, basedate , shift)
```

inmonthtodate
Liefert True, wenn **date** innerhalb des Teils des Monats liegt, der **basedate** enthält, und zwar bis einschließlich der letzten Millisekunde von **basedate**.

```
inmonthtodate (date, basedate , shift)
```

inmonths
Diese Funktion ermittelt, ob ein Zeitstempel im gleichen Monat, Zweimonatszeitraum, Quartal, Viermonatszeitraum oder Halbjahr wie ein Basisdatum liegt. Es lässt sich auch bestimmen, ob ein Zeitstempel in den vorhergehenden oder nachfolgenden Zeitraum fällt.

```
inmonths (n, date, basedate , shift [, first_month_of_year = 1])
```

inmonthstodate
Diese Funktion ermittelt, ob ein Zeitstempel im Teil des Zeitraums von einem Monat, Zweimonatszeitraum, Quartal, Viermonatszeitraum oder Halbjahr liegt, bis einschließlich zur letzten Millisekunde von `base_date`. Es lässt sich auch bestimmen, ob ein Zeitstempel in den vorhergehenden oder nachfolgenden Zeitraum fällt.

```
inmonthstodate (n, date, basedate , shift [, first_month_of_year = 1])
```

inweek
Diese Funktion liefert True, wenn **timestamp** innerhalb der Woche liegt, die **base_date** enthält.

```
inweek (date, basedate , shift [, weekstart])
```

inweektodate
Diese Funktion liefert True, wenn **timestamp** innerhalb des Teils der Woche liegt, der **base_date** enthält, und zwar bis einschließlich der letzten Millisekunde von **base_date**.

```
inweektodate  (date, basedate , shift [, weekstart])
```

inlunarweek

Diese Funktion bestimmt, ob **timestamp** innerhalb der Mondwoche liegt, die **base_date** enthält. Bei Mondwochen in Qlik Sense wird der 1. Januar als der erste Tag der Woche gezählt. Mit Ausnahme der letzten Woche des Jahres umfasst jede Woche genau sieben Tage.

```
inlunarweek  (date, basedate , shift [, weekstart])
```

inlunarweektodate

Diese Funktion gibt an, ob **timestamp** in dem Teil der Mondwoche liegt, die **base_date** enthält, und zwar bis einschließlich der letzten Millisekunde davon. Bei Mondwochen in Qlik Sense wird der 1. Januar als der erste Tag der Woche gezählt. Mit Ausnahme der letzten Woche des Jahres umfasst jede Woche genau sieben Tage.

```
inlunarweektodate  (date, basedate , shift [, weekstart])
```

inday

Diese Funktion liefert True, wenn **timestamp** innerhalb des Tages liegt, der **base_timestamp** enthält.

```
inday (timestamp, basetimestamp , shift [, daystart])
```

indaytotime

Diese Funktion liefert True, wenn **timestamp** in dem Teil des Tages liegt, der **base_timestamp** enthält, und zwar bis auf die Millisekunde von **base_timestamp**.

```
indaytotime (timestamp, basetimestamp , shift [, daystart])
```

Start-... Ende-Funktionen

yearstart

Diese Funktion liefert einen Zeitstempel, der dem Beginn des ersten Jahres mit **date** entspricht. Das Ergebnis wird entsprechend dem im Skript definierten **DateFormat** formatiert.

```
yearstart ( date [, shift = 0 [, first_month_of_year = 1]])
```

yearend

Diese Funktion liefert den Zeitstempel der letzten Millisekunde des letzten Tages des Jahres, in dem **date** liegt. Das Ergebnis wird entsprechend dem im Skript definierten **DateFormat** formatiert.

```
yearend ( date [, shift = 0 [, first_month_of_year = 1]])
```

yearname

Diese Funktion liefert den Zeitstempel der ersten Millisekunde des ersten Tages des Jahres, in dem **date** liegt. Das Ergebnis wird als vierstellige Jahreszahl formatiert.

```
yearname (date [, shift = 0 [, first_month_of_year = 1]] )
```

quarterstart

Diese Funktion liefert den Zeitstempel der ersten Millisekunde des Quartals, in dem **date** liegt. Das Ergebnis wird entsprechend dem im Skript definierten **DateFormat** formatiert.

```
quarterstart (date [, shift = 0 [, first_month_of_year = 1]])
```

quarterend
Diese Funktion liefert den Zeitstempel der letzten Millisekunde des Quartals, in dem **date** liegt. Das Ergebnis wird entsprechend dem im Skript definierten **DateFormat** formatiert.

```
quarterend (date [, shift = 0 [, first_month_of_year = 1]])
```

quartername
Diese Funktion liefert den Zeitstempel der ersten Millisekunde des ersten Tags des Quartals. Das Ergebnis wird als Kombination von Monaten (entsprechend der Skriptvariablen **MonthNames**) und Jahr formatiert.

```
quartername (date [, shift = 0 [, first_month_of_year = 1]])
```

monthstart
Diese Funktion liefert den Zeitstempel der ersten Millisekunde des ersten Tages des Monats, in dem **date** liegt. Das Ergebnis wird entsprechend dem im Skript definierten **DateFormat** formatiert.

```
monthstart (date [, shift = 0])
```

monthend
Diese Funktion liefert den Zeitstempel der letzten Millisekunde des letzten Tags des Monats, in dem `date` liegt. Das Ergebnis wird entsprechend dem im Skript definierten `DateFormat` formatiert.

```
monthend (date [, shift = 0])
```

monthname
Diese Funktion liefert einen Wert mit dem Monat (entsprechend der Skriptvariable **MonthNames** formatiert) und dem Jahr mit einem numerischen Wert, der dem Zeitstempel der ersten Millisekunde des ersten Tags des Monats entspricht.

```
monthname (date [, shift = 0])
```

monthsstart
Diese Funktion gibt einen Wert zurück, der einem Zeitstempel für die erste Millisekunde im Monat, Zweimonatszeitraum, Quartal, Viermonatszeitraum oder Halbjahr entspricht, in dem ein Basisdatum liegt. Es lässt sich auch der Zeitstempel für einen vorhergehenden oder nachfolgenden Zeitraum bestimmen.Das Standardausgabeformat ist das im Skript definierte **DateFormat**.

```
monthsstart (n, date [, shift = 0 [, first_month_of_year = 1]])
```

monthsend
Diese Funktion gibt einen Wert zurück, der einem Zeitstempel für die letzte Millisekunde im Monat, Zweimonatszeitraum, Quartal, Viermonatszeitraum oder Halbjahr entspricht, in dem ein Basisdatum liegt. Es lässt sich auch der Zeitstempel für einen vorhergehenden oder nachfolgenden Zeitraum bestimmen.

```
monthsend (n, date [, shift = 0 [, first_month_of_year = 1]])
```

monthsname
Diese Funktion liefert einen Anzeigewert, der den Bereich der Monate des Zeitraums (formatiert nach der **MonthNames**-Skriptvariable) sowie das Jahr darstellt. Der zugrunde liegende numerische Wert entspricht dem Zeitstempel der ersten Millisekunde des Monats, Zweimonatszeitraums, Quartals, Viermonatszeitraums

oder Halbjahrs, in dem ein Basisdatum liegt.

```
monthsname (n, date [, shift = 0 [, first_month_of_year = 1]])
```

weekstart
Diese Funktion liefert den Zeitstempel der ersten Millisekunde des ersten Tags der Kalenderwoche, in der **date** liegt. Das Ergebnis wird entsprechend dem im Skript definierten **DateFormat** formatiert.

```
weekstart (date [, shift = 0 [,weekoffset = 0]])
```

weekend
Diese Funktion gibt einen Wert zurück, der dem Zeitstempel der letzten Millisekunde des letzten Tages der Kalenderwoche entspricht, in der **date** liegt. Das Ergebnis wird entsprechend dem im Skript definierten **DateFormat** formatiert.

```
weekend (date [, shift = 0 [,weekoffset = 0]])
```

weekname
Diese Funktion liefert den Zeitstempel der ersten Millisekunde der Kalenderwoche, in der **date** liegt. Das Ergebnis wird als Kombination von Jahr und Wochennummer formatiert.

```
weekname (date [, shift = 0 [,weekoffset = 0]])
```

lunarweekstart
Diese Funktion liefert einen Wert, der dem Zeitstempel der ersten Millisekunde des ersten Tags der Mondwoche entspricht, in der **date** liegt. Bei Mondwochen in Qlik Sense wird der 1. Januar als der erste Tag der Woche gezählt. Mit Ausnahme der letzten Woche des Jahres umfasst jede Woche genau sieben Tage.

```
lunarweekstart (date [, shift = 0 [,weekoffset = 0]])
```

lunarweekend
Diese Funktion liefert einen Wert, der dem Zeitstempel der letzten Millisekunde des letzten Tags der Mondwoche entspricht, in der **date** liegt. Bei Mondwochen in Qlik Sense wird der 1. Januar als der erste Tag der Woche gezählt. Mit Ausnahme der letzten Woche des Jahres umfasst jede Woche genau sieben Tage.

```
lunarweekend (date [, shift = 0 [,weekoffset = 0]])
```

lunarweekname
Diese Funktion liefert den Anzeigewert der ersten Millisekunde des ersten Tags der Mondwoche, in der **date** liegt. Das Ergebnis wird als Kombination von Jahr und Mondwochennummer formatiert. Bei Mondwochen in Qlik Sense wird der 1. Januar als der erste Tag der Woche gezählt. Mit Ausnahme der letzten Woche des Jahres umfasst jede Woche genau sieben Tage.

```
lunarweekname (date [, shift = 0 [,weekoffset = 0]])
```

daystart
Diese Funktion liefert den Zeitstempel der ersten Millisekunde des Tages, in dem das Argument **time** liegt. Das Ergebnis wird entsprechend dem im Skript definierten **TimestampFormat** formatiert.

```
daystart (timestamp [, shift = 0 [, dayoffset = 0]])
```

dayend
Diese Skriptfunktion liefert den Zeitstempel der letzten Millisekunde des Tages, in dem **time** liegt. Das Ergebnis wird entsprechend dem im Skript definierten **TimestampFormat** formatiert.

```
dayend (timestamp [, shift = 0 [, dayoffset = 0]])
```

dayname
Diese Funktion liefert den Zeitstempel der ersten Millisekunde des Tages, in dem **time** liegt. Das Ergebnis wird als Datum formatiert.

```
dayname (timestamp [, shift = 0 [, dayoffset = 0]])
```

Funktionen zur Nummerierung der Tage

age
Die Funktion **age** liefert zum Zeitpunkt **timestamp** das Alter (in vollendeten Jahren) einer Person, die am **date_of_birth** geboren ist.

```
age (timestamp, date_of_birth)
```

networkdays
Die Funktion **networkdays** liefert die Zahl der Arbeitstage (Montag bis Freitag) zwischen **start_date** und **end_date**, unter Berücksichtigung eventueller Feiertage unter **holiday**.

```
networkdays  (start:date, end_date {, holiday})
```

firstworkdate
Die Funktion **firstworkdate** liefert das späteste Startdatum zur Vollendung einer gewissen Zahl von **no_of_workdays** (Montag bis Freitag) bis zu einem vorgegebenen **end_date**, unter Berücksichtigung eventueller Feiertage, die als weitere Parameter definiert werden können. **end_date** und **holiday** müssen gültige Datumsangaben oder Zeitstempel sein.

```
firstworkdate (end_date, no_of_workdays {, holiday} )
```

lastworkdate
Die Funktion **lastworkdate** liefert das früheste Enddatum zur Vollendung von **no_of_workdays** (Montag bis Freitag) ab einem vorgegebenen **start_date** unter Berücksichtigung eventueller **holiday**.**start_date** und **holiday** müssen gültige Daten und Zeitstempel sein.

```
lastworkdate (start_date, no_of_workdays {, holiday})
```

daynumberofyear
Diese Funktion berechnet die Nummer des Tages des Jahres, in dem der Zeitstempel liegt. Die Berechnung erfolgt ab der ersten Millisekunde des ersten Tags des Jahres, aber der Beginn des ersten Monats kann festgelegt werden.

```
daynumberofyear (date[,firstmonth])
```

daynumberofquarter

Diese Funktion berechnet die Nummer des Tages des Quartals, in dem der Zeitstempel liegt. Diese Funktion wird verwendet, wenn Sie einen Master-Kalender erstellen.

```
daynumberofquarter (date[,firstmonth])
```

addmonths

Diese Funktion liefert das Datum, das **n** Monate nach **startdate** liegt, bzw. **n** Monate vor **startdate**, wenn **n** negativ ist.

Syntax:

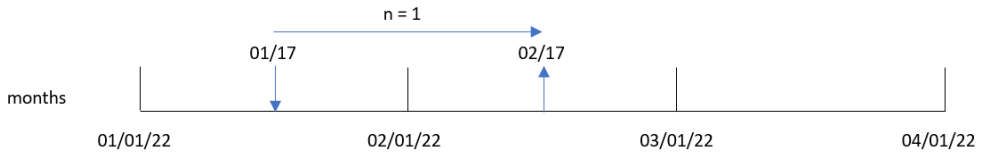
```
AddMonths(startdate, n , [ , mode])
```

Rückgabe Datentyp: dual

Die Funktion `addmonths()` addiert oder subtrahiert eine definierte Anzahl Monate, `n`, von einem `startdate` und gibt das entsprechende Datum zurück.

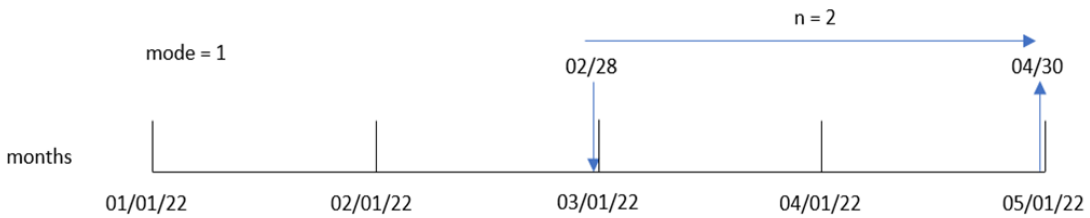
Das Argument `mode` wirkt sich auf die Werte `startdate` an oder nach dem 28. des Monats aus. Wenn das Argument `mode` auf 1 festgelegt ist, gibt die Funktion `addmonths()` ein Datum zurück, dessen relativer Abstand zum Monatsende dem von `startdate` entspricht.

Beispieldiagramm der Funktion `addmonths()`



Beispiel: Der 28. Februar ist der letzte Tag des Monats. Wenn die Funktion `addmonths()` mit einem `mode` von 1 verwendet wird, um das Datum zwei Monate später zurückzugeben, dann gibt die Funktion das letzte Datum des Monats April zurück, den 30. April.

Beispieldiagramm der Funktion `addmonths()` mit `mode=1`



Argumente

Argument	Beschreibung
startdate	Das Startdatum als Zeitstempel, z. B. '2012-10-12'.
n	Monate als positive oder negative ganze Zahl.
mode	Legt fest, ob der Monat entsprechend zum Anfang oder zum Ende des Monats hinzugefügt wird. Der Standardmodus ist 0 für Hinzufügungen entsprechend zum Anfang des Monats. Legen Sie den Modus auf 1 für Hinzufügungen entsprechend zum Ende des Monats fest. Wenn der Modus auf 1 festgelegt und das Eingabedatum der 28. oder höher ist, prüft die Funktion im Startdatum, wie viele Tage bis zum Erreichen des Monatsendes fehlen. Die gleiche Anzahl von Tagen bis zum Monatsende wird für das gelieferte Datum festgelegt.

Verwendung

Die Funktion `addmonths()` wird in der Regel in einer Formel verwendet, um ein Datum zu finden, das eine bestimmte Anzahl Monate vor oder nach einem Zeitraum liegt.

Beispielsweise kann die Funktion `addmonths()` verwendet werden, um das Enddatum von Mobiltelefonverträgen zu identifizieren.

Funktionsbeispiele

Beispiel	Ergebnis
`addmonths ('01/29/2003' ,3)`	Gibt „04/29/2003“ zurück.
`addmonths ('01/29/2003',3,0)`	Gibt „04/29/2003“ zurück.
`addmonths ('01/29/2003',3,1)`	Gibt „04/28/2003“ zurück.
`addmonths ('01/29/2003',1,0)`	Gibt „02/28/2003“ zurück.
`addmonths ('01/29/2003',1,1)`	Gibt „02/26/2003“ zurück.
`addmonths ('02/28/2003',1,0)`	Gibt „03/28/2003“ zurück.
`addmonths ('02/28/2003',1,1)`	Gibt „03/31/2003“ zurück.
`addmonths ('01/29/2003',-3)`	Gibt „10/29/2002“ zurück.

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für

Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Beispiel 1 – keine zusätzlichen Argumente

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz, der eine Reihe von Transaktionen zwischen 2020 und 2022 enthält, wird in eine Tabelle namens `Transactions` geladen.
- Das Datumsfeld wird im Format der Systemvariablen `DateFormat` (MM/TT/JJJJ) bereitgestellt.
- Es wird ein Feld `two_months_later` erstellt, das das Datum zwei Monate nach dem Transaktionsdatum zurückgibt.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        addmonths(date,2) as two_months_later
        ;
Load
*
Inline
[
id,date,amount
8188,'01/10/2020',37.23
8189,'02/28/2020',17.17
8190,'04/09/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'08/14/2020',82.06
8194,'10/07/2020',40.39
8195,'12/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'12/27/2021',25.12
8204,'02/02/2022',46.23
```

```
8205,'02/26/2022',84.21
8206,'03/07/2022',96.24
8207,'03/11/2022',67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

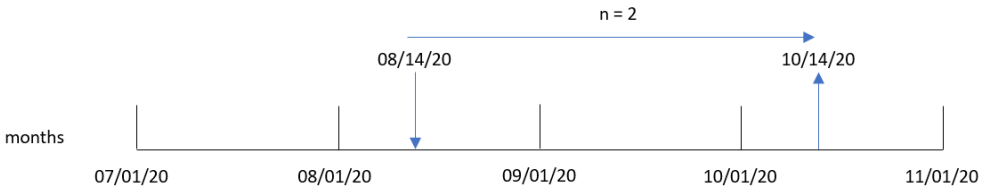
- `date`
- `two_months_later`

Ergebnistabelle

date	two_months_later
01/10/2020	03/10/2020
02/28/2020	04/28/2020
04/09/2020	06/09/2020
04/16/2020	06/16/2020
05/21/2020	07/21/2020
08/14/2020	10/14/2020
10/07/2020	12/07/2020
12/05/2020	02/05/2021
01/22/2021	03/22/2021
02/03/2021	04/03/2021
03/17/2021	05/17/2021
04/23/2021	06/23/2021
05/04/2021	07/04/2021
06/30/2021	08/30/2021
07/26/2021	09/26/2021
12/27/2021	02/27/2022
02/02/2022	04/02/2022
02/26/2022	04/26/2022
03/07/2022	05/07/2022
03/11/2022	05/11/2022

Das Feld „two_months_later“ wird im vorangehenden load-Befehl mithilfe der Funktion addmonths() erstellt. Das erste angegebene Argument identifiziert, welches Datum ausgewertet wird. Das zweite Argument ist die Anzahl der Monate, die zu startdate addiert oder davon abgezogen werden sollen. In diesem Fall wird der Wert 2 angegeben.

Diagramm der Funktion addmonths(), Beispiel ohne zusätzliche Argumente



Transaktion 8193 fand am 14. August statt. Daher gibt die Funktion addmonths() den 14. Oktober 2020 für das Feld two_months_later zurück.

Beispiel 2 – Relatives Monatsende

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz enthält eine Reihe von Monatsende-Transaktionen im Jahr 2022 und wird in eine Tabelle namens Transactions geladen.
- Das Datumsfeld wird im Format der Systemvariablen DateFormat (MM/TT/JJJJ) bereitgestellt.
- Es wird ein Feld relative_two_months_prior erstellt, das das relative Monatsenddatum zwei Monate nach dem Transaktionsdatum zurückgibt.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        addmonths(date,-2,1) as relative_two_months_prior
        ;
Load
*
Inline
[
id,date,amount
8188,'01/28/2022',37.23
8189,'01/31/2022',57.54
8190,'02/28/2022',17.17
8191,'04/29/2022',88.27
```

```
8192,'04/30/2022',57.42
8193,'05/31/2022',53.80
8194,'08/14/2022',82.06
8195,'10/07/2022',40.39
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

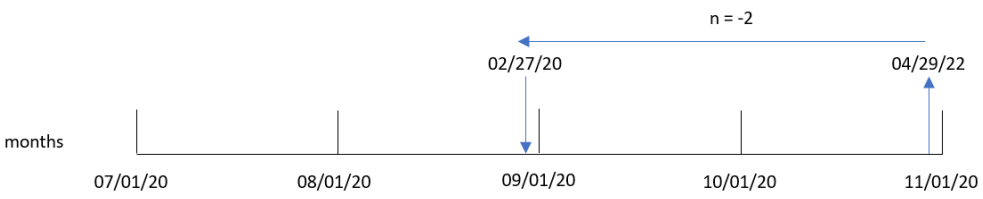
- `date`
- `relative_two_months_prior`

Ergebnistabelle

date	relative_two_months_prior
01/28/2022	11/27/2021
01/31/2022	11/30/2021
02/28/2022	12/31/2021
04/29/2022	02/27/2022
04/30/2022	02/28/2022
05/31/2022	03/31/2022
08/14/2022	06/14/2022
10/07/2022	08/07/2022

Das Feld `relative_two_months_prior` wird in der vorangehenden load-Anweisung mithilfe der Funktion `addmonths()` erstellt. Das erste angegebene Argument identifiziert, welches Datum ausgewertet wird. Das zweite Argument ist die Anzahl der Monate, die zu `startdate` addiert oder davon abgezogen werden sollen. In diesem Fall wird der Wert -2 angegeben. Das letzte Argument ist der Modus mit einem Wert von 1, wodurch erzwungen wird, dass die Funktion das relative Monatsenddatum für alle Datumswerte am oder nach dem 28. berechnet.

Diagramm der Funktion `addmonths()`, Beispiel mit `n=-2`



Transaktion 8191 findet am 29. April 2022 statt. Anfänglich wird der Monat durch „zwei Monate vorher“ auf Februar festgelegt. Da dann das dritte Argument der Funktion den Modus auf 1 festlegt und der Wert des Tages nach dem 27. liegt, berechnet die Funktion den relativen Monatsendwert. Die Funktion identifiziert, dass der 29. der zweitletzte Tag im April ist und gibt daher den zweitletzten Tag für Februar, den 27., zurück.

Beispiel 3 – Diagrammobjektbeispiel

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript verwendet den gleichen Datensatz und das gleiche Szenario wie das erste Beispiel.

In diesem Beispiel wird jedoch der unveränderte Datensatz in die Anwendung geladen. Die Berechnung, die das Datum für zwei Monate nach dem Transaktionsdatum zurückgibt, wird als Kennzahl in einem Diagrammobjekt erstellt.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8188,'01/10/2020',37.23
8189,'02/28/2020',17.17
8190,'04/09/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'08/14/2020',82.06
8194,'10/07/2020',40.39
8195,'12/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'12/27/2021',25.12
8204,'02/02/2022',46.23
8205,'02/26/2022',84.21
8206,'03/07/2022',96.24
8207,'03/11/2022',67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu: `date`.

Erstellen Sie die folgende Kennzahl:

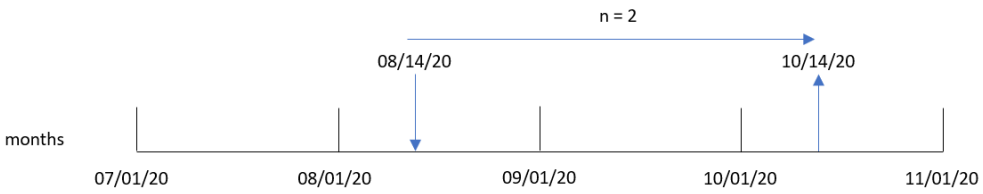
```
=addmonths(date,2)
```

Ergebnistabelle

date	=addmonths(date,2)
01/10/2020	03/10/2020
02/28/2020	04/28/2020
04/09/2020	06/09/2020
04/16/2020	06/16/2020
05/21/2020	07/21/2020
08/14/2020	10/14/2020
10/07/2020	12/07/2020
12/05/2020	02/05/2021
01/22/2021	03/22/2021
02/03/2021	04/03/2021
03/17/2021	05/17/2021
04/23/2021	06/23/2021
05/04/2021	07/04/2021
06/30/2021	08/30/2021
07/26/2021	09/26/2021
12/27/2021	02/27/2022
02/02/2022	04/02/2022
02/26/2022	04/26/2022
03/07/2022	05/07/2022
03/11/2022	05/11/2022

Die Kennzahl `two_months_later` wird im Diagrammobjekt anhand der Funktion `addmonths()` erstellt. Das erste angegebene Argument identifiziert, welches Datum ausgewertet wird. Das zweite Argument ist die Anzahl der Monate, die zu `startdate` addiert oder davon abgezogen werden sollen. In diesem Fall wird der Wert 2 angegeben.

Diagramm der Funktion `addmonths()`, Diagrammobjektbeispiel



Transaktion 8193 fand am 14. August statt. Daher gibt die Funktion `addmonths()` den 14. Oktober 2020 für das Feld `two_months_later` zurück.

Beispiel 4 – Szenario

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Datensatz, der in eine Tabelle namens `Mobile_Plans` geladen wird
- Informationen mit Vertrags-ID, Startdatum, Vertragslänge und monatlicher Gebühr.

Der Endbenutzer möchte ein Diagrammobjekt, das nach Vertrags-ID das Enddatum für jeden Telefonvertrag zurückgibt.

Ladeskript

```
Mobile_Plans:
Load
*
Inline
[
contract_id,start_date,contract_length,monthly_fee
8188,'01/13/2020',18,37.23
8189,'02/26/2020',24,17.17
8190,'03/27/2020',36,88.27
8191,'04/16/2020',24,57.42
8192,'05/21/2020',24,53.80
8193,'08/14/2020',12,82.06
8194,'10/07/2020',18,40.39
8195,'12/05/2020',12,87.21
8196,'01/22/2021',12,95.93
8197,'02/03/2021',18,45.89
8198,'03/17/2021',24,36.23
8199,'04/23/2021',24,25.66
8200,'05/04/2021',12,82.77
8201,'06/30/2021',12,69.98
8202,'07/26/2021',12,76.11
8203,'12/27/2021',36,25.12
8204,'06/06/2022',24,46.23
8205,'07/18/2022',12,84.21
8206,'11/14/2022',12,96.24
8207,'12/12/2022',18,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `contract_id`
- `start_date`
- `contract_length`

Erstellen Sie die folgende Kennzahl, um das Enddatum für jeden Vertrag zu berechnen.

```
=addmonths(start_date,contract_length, 0)
```

Ergebnistabelle

contract_id	start_date	contract_length	=addmonths(start_date,contract_length,0)
8188	01/13/2020	18	07/13/2021
8189	02/26/2020	24	02/26/2022
8190	03/27/2020	36	03/27/2023
8191	04/16/2020	24	04/16/2022
8192	05/21/2020	24	05/21/2022
8193	08/14/2020	12	08/14/2021
8194	10/07/2020	18	04/07/2022
8195	12/05/2020	12	12/05/2021
8196	01/22/2021	12	01/22/2022
8197	02/03/2021	18	08/03/2022
8198	03/17/2021	24	03/17/2023
8199	04/23/2021	24	04/23/2023
8200	05/04/2021	12	05/04/2022
8201	06/30/2021	12	06/30/2022
8202	07/26/2021	12	07/26/2022
8203	12/27/2021	36	12/27/2024
8204	06/06/2022	24	06/06/2024
8205	07/18/2022	12	07/18/2023
8206	11/14/2022	12	11/14/2023
8207	12/12/2022	18	06/12/2024

addyears

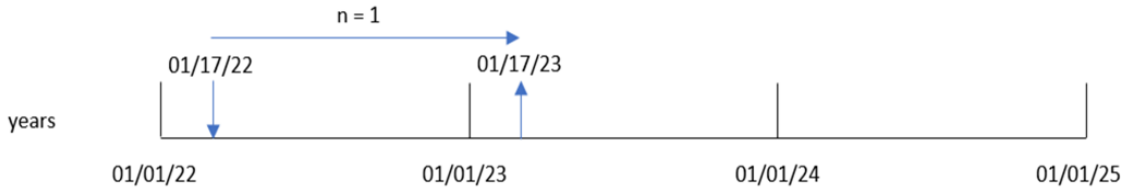
Diese Funktion liefert das Datum, das **n** Jahre nach **startdate** liegt, bzw. **n** Jahre vor **startdate**, wenn **n** negativ ist.

Syntax:

```
AddYears(startdate, n)
```

Rückgabe Datentyp: dual

Beispieldiagramm der Funktion addyears()



Die Funktion addyears() addiert oder subtrahiert eine definierte Anzahl Jahre, n, von einem startdate. Dann gibt sie das resultierende Datum zurück.

Argumente

Argument	Beschreibung
startdate	Das Startdatum als Zeitstempel, z. B. '2012-10-12'.
n	Anzahl Jahre als positive oder negative ganze Zahl.

Funktionsbeispiele

Beispiel	Ergebnis
addyears ('01/29/2010',3)	Gibt „01/29/2013“ zurück.
addyears ('01/29/2010',-1)	Gibt „01/29/2009“ zurück.

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung SET DateFormat in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für

Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Beispiel 1 – Einfaches Beispiel

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz, der eine Reihe von Transaktionen zwischen 2020 und 2022 enthält, wird in eine Tabelle namens `Transactions` geladen.
- Das Datumsfeld wird im Format der Systemvariablen `DateFormat` (MM/TT/JJJJ) bereitgestellt.
- Es wird ein Feld `two_years_later` erstellt, das das Datum zwei Jahre nach dem Transaktionsdatum zurückgibt.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        addyears(date,2) as two_years_later
        ;
Load
*
Inline
[
id,date,amount
8188,'01/10/2020',37.23
8189,'02/28/2020',17.17
8190,'04/09/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'08/14/2020',82.06
8194,'10/07/2020',40.39
8195,'12/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'12/27/2021',25.12
8204,'02/02/2022',46.23
```

```
8205,'02/26/2022',84.21
8206,'03/07/2022',96.24
8207,'03/11/2022',67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

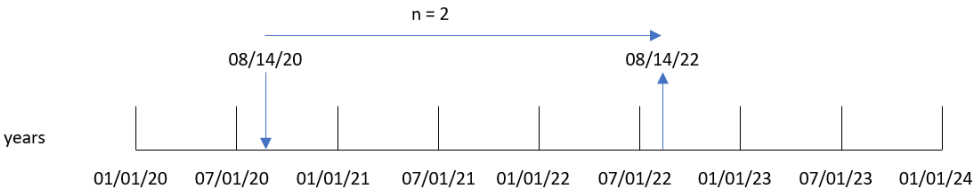
- `date`
- `two_years_later`

Ergebnistabelle

date	two_years_later
01/10/2020	01/10/2022
02/28/2020	02/28/2022
04/09/2020	04/09/2022
04/16/2020	04/16/2022
05/21/2020	05/21/2022
08/14/2020	08/14/2022
10/07/2020	10/07/2022
12/05/2020	12/05/2022
01/22/2021	01/22/2023
02/03/2021	02/03/2023
03/17/2021	03/17/2023
04/23/2021	04/23/2023
05/04/2021	05/04/2023
06/30/2021	06/30/2023
07/26/2021	07/26/2023
12/27/2021	12/27/2023
02/02/2022	02/02/2024
02/26/2022	02/26/2024
03/07/2022	03/07/2024
03/11/2022	03/11/2024

Das Feld „two_years_later“ wird im vorangehenden load-Befehl mithilfe der Funktion addyears() erstellt. Das erste angegebene Argument identifiziert, welches Datum ausgewertet wird. Das zweite Argument ist die Anzahl der Jahre, die zum Startdatum addiert oder davon abgezogen werden sollen. In diesem Fall wird der Wert 2 angegeben.

Diagramm der Funktion addyears(), einfaches Beispiel



Transaktion 8193 fand am 14. August 2020 statt. Daher gibt die Funktion addyears() den 14. August 2022 für das Feld two_years_later zurück.

Beispiel 2 – Diagrammobjektbeispiel

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz, der eine Reihe von Transaktionen zwischen 2020 und 2022 enthält, wird in eine Tabelle namens Transactions geladen.
- Das Datumsfeld wird im Format der Systemvariablen DateFormat (MM/TT/JJJJ) bereitgestellt.

Erstellen Sie in einem Diagrammobjekt eine Kennzahl prior_year_date, die das Datum ein Jahr vor dem Transaktionsdatum zurückgibt.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8188,'01/10/2020',37.23
8189,'02/28/2020',17.17
8190,'04/09/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'08/14/2020',82.06
```

```
8194,'10/07/2020',40.39
8195,'12/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'12/27/2021',25.12
8204,'02/02/2022',46.23
8205,'02/26/2022',84.21
8206,'03/07/2022',96.24
8207,'03/11/2022',67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu: `date`.

Erstellen Sie die folgende Kennzahl, um das Datum ein Jahr vor jeder Transaktion zu berechnen:

```
=addyears(date,-1)
```

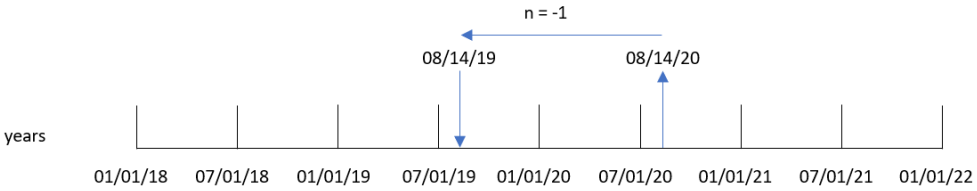
Ergebnistabelle

date	=addyears(date,-1)
01/10/2020	01/10/2019
02/28/2020	02/28/2019
04/09/2020	04/09/2019
04/16/2020	04/16/2019
05/21/2020	05/21/2019
08/14/2020	08/14/2019
10/07/2020	10/07/2019
12/05/2020	12/05/2019
01/22/2021	01/22/2020
02/03/2021	02/03/2020
03/17/2021	03/17/2020
04/23/2021	04/23/2020
05/04/2021	05/04/2020
06/30/2021	06/30/2020
07/26/2021	07/26/2020

date	=addyears(date,-1)
12/27/2021	12/27/2020
02/02/2022	02/02/2021
02/26/2022	02/26/2021
03/07/2022	03/07/2021
03/11/2022	03/11/2021

Die Kennzahl `one_year_prior` wird im Diagrammobjekt anhand der Funktion `addyears()` erstellt. Das erste angegebene Argument identifiziert, welches Datum ausgewertet wird. Das zweite Argument ist die Anzahl der Jahre, die zum `startdate` addiert oder davon abgezogen werden sollen. In diesem Fall wird der Wert -1 angegeben.

Diagramm der Funktion `addyears()`, Diagrammobjektbeispiel



Transaktion 8193 fand am 14. August statt. Daher gibt die Funktion `addyears()` den 14. August 2019 für das Feld `one_year_prior` zurück.

Beispiel 3 – Szenario

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Datensatz, der in eine Tabelle namens `Warranties` geladen wird
- Informationen mit Produkt-ID, Kaufdatum, Garantiedauer und Kaufpreis.

Der Endbenutzer möchte ein Diagrammobjekt, das nach Produkt-ID das Garantieenddatum für jedes Produkt anzeigt.

Ladeskript

```
Warranties:
Load
*
Inline
[
```

```
product_id,purchase_date,warranty_length,purchase_price
8188,'01/13/2020',4,32000
8189,'02/26/2020',2,28000
8190,'03/27/2020',3,41000
8191,'04/16/2020',4,17000
8192,'05/21/2020',2,25000
8193,'08/14/2020',1,59000
8194,'10/07/2020',2,12000
8195,'12/05/2020',3,12000
8196,'01/22/2021',4,24000
8197,'02/03/2021',1,50000
8198,'03/17/2021',2,80000
8199,'04/23/2021',3,10000
8200,'05/04/2021',4,30000
8201,'06/30/2021',3,30000
8202,'07/26/2021',4,20000
8203,'12/27/2021',4,10000
8204,'06/06/2022',2,25000
8205,'07/18/2022',1,32000
8206,'11/14/2022',1,30000
8207,'12/12/2022',4,22000
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `product_id`
- `purchase_date`
- `warranty_length`

Erstellen Sie die folgende Kennzahl, um das Enddatum für jede Produktgarantie zu berechnen.

```
=addyears(purchase_date,warranty_length)
```

Ergebnistabelle

product_id	purchase_date	warranty_length	=addyears(purchase_date,warranty_length)
8188	01/13/2020	4	01/13/2024
8189	02/26/2020	2	02/26/2022
8190	03/27/2020	3	03/27/2023
8191	04/16/2020	4	04/16/2024
8192	05/21/2020	2	05/21/2022
8193	08/14/2020	1	08/14/2021
8194	10/07/2020	2	10/07/2022
8195	12/05/2020	3	12/05/2023

product_id	purchase_date	warranty_length	=addyears(purchase_date,warranty_length)
8196	01/22/2021	4	01/22/2025
8197	02/03/2021	1	02/03/2022
8198	03/17/2021	2	03/17/2023
8199	04/23/2021	3	04/23/2024
8200	05/04/2021	4	05/04/2025
8201	06/30/2021	3	06/30/2024
8202	07/26/2021	4	07/26/2025
8203	12/27/2021	4	12/27/2025
8204	06/06/2022	2	06/06/2024
8205	07/18/2022	1	07/18/2023
8206	11/14/2022	1	11/14/2023
8207	12/12/2022	4	12/12/2026

age

Die Funktion **age** liefert zum Zeitpunkt **timestamp** das Alter (in vollendeten Jahren) einer Person, die am **date_of_birth** geboren ist.

Syntax:

```
age(timestamp, date_of_birth)
```

Kann eine Formel sein.

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
timestamp	Der Zeitstempel oder die Formel, die einen Zeitstempel ergibt, bis zu dem/der die Anzahl der vollendeten Jahre berechnet werden soll.
date_of_birth	Geburtsdatum der Person, deren Alter berechnet wird. Kann eine Formel sein.

Beispiele und Ergebnisse:
In diesen Beispielen wird das Datumsformat **DD/MM/YYYY** verwendet. Das Datumsformat wird im Befehl **SET DateFormat** oben in Ihrem Datenladeskript angegeben. Ändern Sie das Format in den Beispielen nach Bedarf.

Skriptbeispiele

Beispiel	Ergebnis
`age('25/01/2014', '29/10/2012')`	Liefert 1.
`age('29/10/2014', '29/10/2012')`	Liefert 2.

Beispiel:

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Fügen Sie einem Arbeitsblatt in Ihrer App dann die Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um das Ergebnis anzuzeigen.

```
Employees:
LOAD * INLINE [
Member|DateOfBirth
John|28/03/1989
Linda|10/12/1990
Steve|5/2/1992
Birg|31/3/1993
Raj|19/5/1994
Prita|15/9/1994
Su|11/12/1994
Goran|2/3/1995
Sunny|14/5/1996
Ajoa|13/6/1996
Daphne|7/7/1998
Biffy|4/8/2000
] (delimiter is |);
AgeTable:
Load *,
age('20/08/2015', DateOfBirth) As Age
Resident Employees;
Drop table Employees;
```

Die entstehende Tabelle enthält die zurückgegebenen Werte von age für jeden der Datensätze in der Tabelle.

Ergebnistabelle

Member	DateOfBirth	Age
John	28/03/1989	26
Linda	10/12/1990	24
Steve	5/2/1992	23
Birg	31/3/1993	22
Raj	19/5/1994	21
Prita	15/9/1994	20
Su	11/12/1994	20
Goran	2/3/1995	20

Member	DateOfBirth	Age
Sunny	14/5/1996	19
Ajoa	13/6/1996	19
Daphne	7/7/1998	17
Biffy	4/8/2000	15

converttolocaltime

Konvertiert einen UTC- oder GMT-Zeitstempel in eine lokale Zeit als dualen Wert. Der Standort kann eine beliebige Stadt und Ortsbezeichnung in jeder Zeitzone weltweit sein.

Syntax:

```
ConvertToLocalTime(timestamp [, place [, ignore_dst=false]])
```

Rückgabe Datentyp: dual

Argumente

Argument	Beschreibung
timestamp	Der Zeitstempel bzw. eine Formel zur Berechnung eines Zeitstempels zur Umwandlung.

Argument	Beschreibung
place	Ein Ort oder eine Zeitzone aus der nachfolgenden Tabelle der gültigen Orte und Zeitzonen. Alternativ können Sie GMT oder UTC zur Definition einer Ortszeit verwenden. Die folgenden Werte und Zeitverschiebungen sind gültig: • GMT • GMT-12:00 - GMT-01:00 • GMT+01:00 - GMT+14:00 • UTC • UTC-12:00 - UTC-01:00 • UTC+01:00 - UTC+14:00 *Falls Sie eine DST-Verschiebung verwenden (Sie also einen Argumentwert **ignore_dst** angeben, der den Wert False setzt), müssen Sie einen Ort anstatt einer GMT-Verschiebung im **place**-Argument angeben. Dies beruht darauf, dass für die Anpassung der Sommerzeit zusätzlich zu den Längengradinformationen, die durch eine GMT-Verschiebung bereitgestellt werden, Breitengradinformationen erforderlich sind. Informationen finden Sie unter Verwenden von GMT-Verschiebungen in Verbindung mit DST (page 642).* *Sie können nur die Standardzeitverschiebungen verwenden. Es ist nicht möglich, beliebige Zeitverschiebungen anzugeben, zum Beispiel GMT-04:27.*
ignore_dst	Falls dieses Argument den Wert True setzt, wird DST (Sommerzeit) ignoriert. Gültige Argumentwerte, die auf True gesetzt werden, enthalten `-1` und `True()`. Falls dieses Argument den Wert auf False setzt, wird der Zeitstempel für die Sommerzeit angepasst. Gültige Argumentwerte, die auf False gesetzt werden, enthalten `0` und `False()`. Falls das Argument **ignore_dst** ungültig ist, wertet die Funktion den Ausdruck aus, als ob der Wert **ignore_dst** den Wert True festlegt. Falls das Argument **ignore_dst** nicht angegeben wird, wertet die Funktion den Ausdruck aus, als ob der Wert **ignore_dst** den Wert False festlegt.

Gültige Orte und Zeitzonen

A-C	D-K	L-R	S-Z
Abu Dhabi	Darwin	La Paz	Samoa
Adelaide	Dhaka	Lima	Santiago
Alaska	Eastern Time (US & Canada)	Lisbon	Sapporo
Amsterdam	Edinburgh	Ljubljana	Sarajevo

A-C	D-K	L-R	S-Z
Arizona	Ekaterinburg	London	Saskatchewan
Astana	Fiji	Madrid	Seoul
Athens	Georgetown	Magadan	Singapore
Atlantic Time (Canada)	Greenland	Mazatlan	Skopje
Auckland	Greenwich Mean Time : Dublin	Melbourne	Sofia
Azores	Guadalajara	Mexico City	Solomon Is.
Baghdad	Guam	Mid-Atlantic	Sri Jayawardenepura
Baku	Hanoi	Minsk	St. Petersburg
Bangkok	Harare	Monrovia	Stockholm
Beijing	Hawaii	Monterrey	Sydney
Belgrade	Helsinki	Moscow	Taipei
Berlin	Hobart	Mountain Time (US & Canada)	Tallinn
Bern	Hong Kong	Mumbai	Tashkent
Bogota	Indiana (East)	Muscat	Tbilisi
Brasilia	International Date Line West	Nairobi	Tehran
Bratislava	Irkutsk	New Caledonia	Tokyo
Brisbane	Islamabad	New Delhi	Urumqi
Brussels	Istanbul	Newfoundland	Warsaw
Bucharest	Jakarta	Novosibirsk	Wellington
Budapest	Jerusalem	Nuku'alofa	West Central Africa
Buenos Aires	Kabul	Osaka	Vienna
Cairo	Kamchatka	Pacific Time (US & Canada)	Vilnius
Canberra	Karachi	Paris	Vladivostok
Cape Verde Is.	Kathmandu	Perth	Volgograd
Caracas	Kolkata	Port Moresby	Yakutsk
Casablanca	Krasnoyarsk	Prague	Yerevan
Central America	Kuala Lumpur	Pretoria	Zagreb

A-C	D-K	L-R	S-Z
Central Time (US & Canada)	Kuwait	Quito	-
Chennai	Kyiv	Riga	-
Chihuahua	-	Riyadh	-
Chongqing	-	Rome	-
Copenhagen	-	-	-

Beispiele und Ergebnisse:

Skriptbeispiele

Beispiel	Ergebnis
`ConvertToLocalTime('2023-08-14 08:39:47','Paris')`	Gibt '2023-08-14 10:39:47' und den zugehörigen internen Zeitstempel zurück.
`ConvertToLocalTime(UTC(), 'Stockholm')`	Gibt die Uhrzeit für Stockholm zurück und berücksichtigt dabei die Sommerzeit.
`ConvertToLocalTime(UTC(), 'Stockholm', -1)`	Gibt die Uhrzeit für Stockholm zurück und berücksichtigt dabei nicht die Sommerzeit.
`ConvertToLocalTime(UTC(), 'GMT-05:00')`	Gibt die Uhrzeit für die nordamerikanische Ostküste zurück, zum Beispiel New York. Es wird keine Anpassung für die Sommerzeit vorgenommen, da eine GMT-Verschiebung und kein Ort angegeben ist.
`ConvertToLocalTime(UTC(), 'New York', -1)`	Gibt die Uhrzeit für die nordamerikanische Ostküste (New York) zurück, ohne dass die Umstellung der Sommerzeit berücksichtigt wird.
`ConvertToLocalTime(UTC(), 'New York', True())`	Gibt die Uhrzeit für die nordamerikanische Ostküste (New York) zurück, ohne dass die Umstellung der Sommerzeit berücksichtigt wird.
`ConvertToLocalTime(UTC(), 'New York', 0)`	Liefert die Uhrzeit für die nordamerikanische Ostküste (New York), wobei die Umstellung der Sommerzeit berücksichtigt wird.
`ConvertToLocalTime(UTC(), 'New York', False())`	Liefert die Uhrzeit für die nordamerikanische Ostküste (New York), wobei die Umstellung der Sommerzeit berücksichtigt wird.

Verwenden von GMT-Verschiebungen in Verbindung mit DST

Anhand der Implementierung der ICU-Bibliotheken (International Components for Unicode in Qlik Sense erfordert die Nutzung von GMT-Verschiebungen (Greenwich Mean Time) in Kombination mit DST (Sommerzeit) zusätzliche Breitengradinformationen.

GMT ist eine Breitengradverschiebung (Ost-West), wohingegen DST eine Längengradverschiebung (Nord-Süd) ist. Helsinki (Finnland) und Johannesburg (Südafrika) haben dieselbe Verschiebung von GMT+02:00, allerdings nicht dieselbe DST-Verschiebung. Dies bedeutet, dass eine beliebige DST-Verschiebung außer der GMT-Verschiebung Informationen zur Breitengradposition der lokalen Zeitzone (geografische Zeitzoneneingabe) benötigt, damit vollständige Informationen über lokale DST-Bedingungen vorliegen.

day

Diese Funktion liefert den Tag als ganze Zahl, wenn **expression** entsprechend dem Standardformat als Datum interpretiert wird.

Die Funktion gibt den Tag des Monats für eine bestimmtes Datum zurück. Sie wird gewöhnlich verwendet, um ein Tagesfeld als Teil einer Kalenderdimension abzuleiten.

Syntax:

```
day(expression)
```

Rückgabe Datentyp: ganze Zahl

Funktionsbeispiele

Beispiel	Ergebnis
`day( 1971-10-12 )`	liefert 12
`day( 35648 )`	liefert 6, da 35648 = 1997-08-06

Beispiel 1 – DateFormat-Datensatz (Skript)

Ladeskript und Ergebnisse

Überblick

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Einen Datensatz mit Datumsangaben mit dem Namen `Master_Calendar`. Die Systemvariable `DateFormat` wird auf TT/MM/JJJJ festgelegt.
- Einen vorangehenden load-Befehl, mit dem unter Verwendung der Funktion `day()` ein zusätzliches Feld mit dem Namen `day_of_month` erstellt wird.
- Ein weiteres Feld mit dem Namen `long_date`, das die Funktion `date()` verwendet, um den vollständigen Namen des Monats auszudrücken.

Ladeskript

```
SET DateFormat='DD/MM/YYYY';

Master_Calendar:
Load
        date,
        date(date,'dd-MMMM-YYYY') as long_date,
        day(date) as day_of_month
Inline
[
date
03/11/2022
03/12/2022
03/13/2022
03/14/2022
03/15/2022
03/16/2022
03/17/2022
03/18/2022
03/19/2022
03/20/2022
03/21/2022
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `long_date`
- `day_of_month`

Ergebnistabelle

date	long_date	day_of_month
03/11/2022	11-March- 2022	11
03/12/2022	12-March- 2022	12
03/13/2022	13-March- 2022	13
03/14/2022	14-March- 2022	14
03/15/2022	15-March- 2022	15
03/16/2022	16-March- 2022	16
03/17/2022	17-March- 2022	17
03/18/2022	18-March- 2022	18
03/19/2022	19-March- 2022	19

date	long_date	day_of_month
03/20/2022	20-March- 2022	20
03/21/2022	21-March- 2022	21

Der Tag des Monats wird korrekt von der Funktion `day()` im Skript ausgewertet.

Beispiel 2 – ANSI-Datum (Skript)

Ladeskript und Ergebnisse

Überblick

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Einen Datensatz mit Datumsangaben mit dem Namen `Master_Calendar`. Die DateFormat-Systemvariable TT/MM/JJJJ wird verwendet. Die im Datensatz enthaltenen Datumsangaben weisen aber das ANSI-Standarddatumsformat auf.
- Einen vorangehenden load-Befehl, mit dem ein zusätzliches Feld mit dem Namen `day_of_month` erstellt wird, unter Verwendung der Funktion `date()`.
- Ein weiteres Feld mit dem Namen `long_date`, das die Funktion `date()` verwendet, um das Datum mit dem vollständigen Namen des Monats auszudrücken.

Ladeskript

```
SET DateFormat='DD/MM/YYYY';
Master_Calendar:
Load
       date,
       date(date,'dd-MMMM-YYYY') as long_date,
       day(date) as day_of_month

Inline
[
date
2022-03-11
2022-03-12
2022-03-13
2022-03-14
2022-03-15
2022-03-16
2022-03-17
2022-03-18
2022-03-19
2022-03-20
2022-03-21
 ];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `long_date`
- `day_of_month`

Ergebnistabelle

date	long_date	day_of_month
03/11/2022	11-March- 2022	11
03/12/2022	12-March- 2022	12
03/13/2022	13-March- 2022	13
03/14/2022	14-March- 2022	14
03/15/2022	15-March- 2022	15
03/16/2022	16-March- 2022	16
03/17/2022	17-March- 2022	17
03/18/2022	18-March- 2022	18
03/19/2022	19-March- 2022	19
03/20/2022	20-March- 2022	20
03/21/2022	21-March- 2022	21

Der Tag des Monats wird korrekt von der Funktion `day()` im Skript ausgewertet.

Beispiel 3 – Unformatiertes Datum (Skript)

Ladeskript und Ergebnisse

Überblick

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Einen Datensatz mit Datumsangaben mit dem Namen `Master_Calendar`. Die `DateFormat`-Systemvariable TT/MM/JJJJ wird verwendet.
- Einen vorangehenden load-Befehl, mit dem ein zusätzliches Feld mit dem Namen `day_of_month` erstellt wird, unter Verwendung der Funktion `day()`.
- Das ursprüngliche unformatierte Datum mit dem Namen `unformatted_date`.

- Ein weiteres Feld mit dem Namen `long_date`, das `date()` verwendet, wird genutzt, um das numerische Datum in ein formatiertes Datumsfeld zu konvertieren.

Ladeskript

```
SET DateFormat='DD/MM/YYYY';

Master_Calendar:
Load
      unformatted_date,
      date(unformatted_date,'dd-MMMM-YYYY') as long_date,
      day(date) as day_of_month

Inline
[
unformatted_date
44868
44898
44928
44958
44988
45018
45048
45078
45008
45038
45068
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `unformatted_date`
- `long_date`
- `day_of_month`

Ergebnistabelle

unformatted_date	long_date	day_of_month
44868	03-November- 2022	3
44898	03-December- 2022	3
44928	02-January- 2023	2
44958	01-February- 2023	1
44988	03-March- 2023	3

unformatted_date	long_date	day_of_month
45008	23-March- 2023	23
45018	02-April- 2023	2
45038	22-April- 2023	22
45048	02-May- 2023	2
45068	22-May- 2023	22
45078	01-June- 2023	1

Der Tag des Monats wird korrekt von der Funktion `day()` im Skript ausgewertet.

Beispiel 4 – Berechnen des Ablaufmonats (Diagramm)

Ladeskript und Diagrammformel

Überblick

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz der im März aufgegebenen Bestellungen mit dem Namen `orders`. Die Tabelle enthält drei Felder:
    - id
    - order_date
    - amount

Ladeskript

```
Orders:
Load
        id,
        order_date,
        amount
Inline
[
id,order_date,amount
1,03/01/2022,231.24
2,03/02/2022,567.28
3,03/03/2022,364.28
4,03/04/2022,575.76
5,03/05/2022,638.68
6,03/06/2022,785.38
7,03/07/2022,967.46
8,03/08/2022,287.67
9,03/09/2022,764.45
10,03/10/2022,875.43
11,03/11/2022,957.35
```

```
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu:`order_date`.

Erstellen Sie die folgende Kennzahl, um das Lieferdatum zu berechnen: `=day(order_date+5)`.

Ergebnistabelle

order_date	=day(order_date+5)
03/11/2022	16
03/12/2022	17
03/13/2022	18
03/14/2022	19
03/15/2022	20
03/16/2022	21
03/17/2022	22
03/18/2022	23
03/19/2022	24
03/20/2022	25
03/21/2022	26

Die Funktion `day()` berechnet korrekt, dass eine am 11. März aufgegebene Bestellung aufgrund des 5-Tages-Lieferzeitraums am 16. geliefert wird.

dayend

Diese Skriptfunktion liefert den Zeitstempel der letzten Millisekunde des Tages, in dem **time** liegt. Das Ergebnis wird entsprechend dem im Skript definierten **TimestampFormat** formatiert.

Syntax:

```
DayEnd(time[, [period_no[, day_start]])
```

Verwendung

Die Funktion `dayend()` wird in der Regel als Teil einer Formel verwendet, wenn in der Berechnung der Teil des Tages verwendet werden soll, der noch nicht eingetreten ist. Damit können beispielsweise die Gesamtausgaben berechnet werden, die im Lauf des Tages noch anfallen werden.

Rückgabe Datentyp: dual

Argumente

Argument	Beschreibung
time	Der zu berechnende Zeitstempel.
period_no	**period_no** ist eine ganze Zahl oder eine Formel, die eine ganze Zahl ergibt, wobei 0 für den Tag steht, der **time** enthält. Negative Werte von **period_no** stehen für vorangehende, positive Werte für nachfolgende Tage.
day_start	Um anzugeben, dass Tage nicht um Mitternacht beginnen, geben Sie einen Versatz als Bruchteil eines Tages in **day_start** an. So steht beispielsweise 0,125 für 3:00 Uhr. Um den Versatz zu erstellen, teilen Sie also die Startuhrzeit durch 24 Stunden. Damit beispielsweise ein Tag um 7:00 Uhr beginnt, verwenden Sie den Bruch 7/24.

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Funktionsbeispiele

Beispiel	Ergebnis
dayend('01/25/2013 16:45:00')	Liefert 01/25/2013 23:59:59. PM
dayend('01/25/2013 16:45:00', -1)	Liefert 01/24/2013 23:59:59. PM
dayend('01/25/2013 16:45:00', 0, 0.5)	Liefert 01/26/2013 11:59:59. PM

Beispiel 1 – Basisskript

Ladeskript und Ergebnisse

Überblick

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Einen Datensatz, der eine Liste der Datumsangaben enthält, die in eine Tabelle mit dem Namen „Calendar“ geladen werden.
- Die DateFormat-Standardsystemvariable (MM/DD/YYYY).
- Einen vorangehenden load-Befehl zum Erstellen eines zusätzlichen Feldes mit dem Namen „EOD_timestamp“, unter Verwendung der Funktion dayend().

Ladeskript

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Calendar:
    Load
      date,
      dayend(date) as EOD_timestamp
      ;
Load
date
Inline
[
date
03/11/2022 1:47:15 AM
03/12/2022 4:34:58 AM
03/13/2022 5:15:55 AM
03/14/2022 9:25:14 AM
03/15/2022 10:06:54 AM
03/16/2022 10:44:42 AM
03/17/2022 11:33:30 AM
03/18/2022 12:58:14 PM
03/19/2022 4:23:12 PM
03/20/2022 6:42:15 PM
03/21/2022 7:41:16 PM
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- date
- EOD_timestamp

Ergebnistabelle

date	EOD_timestamp
03/11/2022 1:47:15 AM	3/11/2022 11:59:59 PM
03/12/2022 4:34:58 AM	3/12/2022 11:59:59 PM
03/13/2022 5:15:55 AM	3/13/2022 11:59:59 PM
03/14/2022 9:25:14 AM	3/14/2022 11:59:59 PM

date	EOD_timestamp
03/15/2022 10:06:54 AM	3/15/2022 11:59:59 PM
03/16/2022 10:44:42 AM	3/16/2022 11:59:59 PM
03/17/2022 11:33:30 AM	3/17/2022 11:59:59 PM
03/18/2022 12:58:14 PM	3/18/2022 11:59:59 PM
03/19/2022 4:23:12 PM	3/19/2022 11:59:59 PM
03/20/2022 6:42:15 PM	3/20/2022 11:59:59 PM
03/21/2022 7:41:16 PM	3/21/2022 11:59:59 PM

Wie Sie in der Tabelle oben sehen können, wird der Tagesende-Zeitstempel für jedes Datum in unserem Datensatz erstellt. Der Zeitstempel hat das Format der Systemvariablen `TimestampFormat M/D/YYYY h:mm:ss [.fff] TT`.

Beispiel 2 – period_no

Ladeskript und Ergebnisse

Überblick

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Sie laden einen Datensatz mit Dienstbuchungen in eine Tabelle mit dem Namen „Services“.

Der Datensatz umfasst die folgenden Felder:

- `service_id`
- `service_date`
- `amount`

Sie erstellen zwei neue Felder in der Tabelle:

- `deposit_due_date`: Das Datum, an dem die Anzahlung eingehen sollte. Dies ist das Tagesende drei Tage vor dem `service_date`.
- `final_payment_due_date`: Das Datum, an dem die abschließende Zahlung eingehen sollte. Dies ist das Tagesende sieben Tage nach dem `service_date`.

Die zwei obigen Felder werden in einer vorangehenden load-Anweisung mit der Funktion `dayend()` erstellt, und sie liefern die ersten beiden Parameter, `time` und `period_no`.

Ladeskript

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Services:
   Load
     *,
     dayend(service_date,-3) as deposit_due_date,
```

```
     dayend(service_date,7) as final_payment_due_date
     ;
Load
service_id,
service_date,
amount
Inline
[
service_id, service_date,amount
1,03/11/2022 9:25:14 AM,231.24
2,03/12/2022 10:06:54 AM,567.28
3,03/13/2022 10:44:42 AM,364.28
4,03/14/2022 11:33:30 AM,575.76
5,03/15/2022 12:58:14 PM,638.68
6,03/16/2022 4:23:12 PM,785.38
7,03/17/2022 6:42:15 PM,967.46
8,03/18/2022 7:41:16 PM,287.67
9,03/19/2022 8:14:15 PM,764.45
10,03/20/2022 9:23:51 PM,875.43
11,03/21/2022 10:04:41 PM,957.35
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- service_date
- deposit_due_date
- final_payment_due_date

Ergebnistabelle

service_date	deposit_due_date	final_payment_due_date
03/11/2022 9:25:14 AM	3/8/2022 11:59:59 PM	3/18/2022 11:59:59 PM
03/12/2022 10:06:54 AM	3/9/2022 11:59:59 PM	3/19/2022 11:59:59 PM
03/13/2022 10:44:42 AM	3/10/2022 11:59:59 PM	3/20/2022 11:59:59 PM
03/14/2022 11:33:30 AM	3/11/2022 11:59:59 PM	3/21/2022 11:59:59 PM
03/15/2022 12:58:14 PM	3/12/2022 11:59:59 PM	3/22/2022 11:59:59 PM
03/16/2022 4:23:12 PM	3/13/2022 11:59:59 PM	3/23/2022 11:59:59 PM
03/17/2022 6:42:15 PM	3/14/2022 11:59:59 PM	3/24/2022 11:59:59 PM
03/18/2022 7:41:16 PM	3/15/2022 11:59:59 PM	3/25/2022 11:59:59 PM
03/19/2022 8:14:15 PM	3/16/2022 11:59:59 PM	3/26/2022 11:59:59 PM
03/20/2022 9:23:51 PM	3/17/2022 11:59:59 PM	3/27/2022 11:59:59 PM
03/21/2022 10:04:41 PM	3/18/2022 11:59:59 PM	3/28/2022 11:59:59 PM

Die Werte der neuen Felder sind im `TimestampFormat M/D/YYYY h:mm:ss[.fff] TT`. Da die Funktion `dayend ()` verwendet wurde, sind die Zeitstempelwerte alle die letzte Millisekunde des Tages.

Die Werte für die Fälligkeit der Anzahlung sind drei Tage vor dem Fälligkeitsdatum des Dienstes, da das zweite Argument, das in der Funktion `dayend()` übergeben wird, negativ ist.

Die Werte für die Fälligkeit der abschließenden Zahlung sind sieben Tage nach dem Fälligkeitsdatum des Dienstes, da das zweite Argument, das in der Funktion `dayend()` übergeben wird, positiv ist.

Beispiel 3 – day_start script

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Der Datensatz und das Szenario in diesem Beispiel sind die Gleichen wie im vorherigen Beispiel.

Wie im vorherigen Beispiel erstellen Sie zwei neue Felder:

- `deposit_due_date`: Das Datum, an dem die Anzahlung eingehen sollte. Dies ist das Tagesende drei Tage vor dem `service_date`.
- `final_payment_due_date`: Das Datum, an dem die abschließende Zahlung eingehen sollte. Dies ist das Tagesende sieben Tage nach dem `service_date`.

Ihr Unternehmen möchte aber mit einer Richtlinie arbeiten, bei der der Arbeitstag um 5 Uhr nachmittags beginnt und um 5 Uhr nachmittags am Folgetag endet. Ihr Unternehmen kann dann Transaktionen überwachen, die während dieser Arbeitsstunden stattgefunden haben.

Um diese Anforderungen zu erfüllen, werden die zwei obigen Felder in einer vorangehenden load-Anweisung mit der Funktion `dayend()` erstellt und verwenden alle drei Argumente, `time`, `period_no` und `day_start`.

Ladeskript

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Services:
   Load
     *,
     dayend(service_date,-3,17/24) as deposit_due_date,
     dayend(service_date,7,17/24) as final_payment_due_date
     ;
Load
service_id,
service_date,
amount
Inline
[
service_id, service_date,amount
1,03/11/2022 9:25:14 AM,231.24
2,03/12/2022 10:06:54 AM,567.28
3,03/13/2022 10:44:42 AM,364.28
```

```
4,03/14/2022 11:33:30 AM,575.76
5,03/15/2022 12:58:14 PM,638.68
6,03/16/2022 4:23:12 PM,785.38
7,03/17/2022 6:42:15 PM,967.46
8,03/18/2022 7:41:16 PM,287.67
9,03/19/2022 8:14:15 PM,764.45
10,03/20/2022 9:23:51 PM,875.43
11,03/21/2022 10:04:41 PM,957.35
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `service_date`
- `deposit_due_date`
- `final_payment_due_date`

Ergebnistabelle

service_date	deposit_due_date	final_payment_due_date
03/11/2022 9:25:14 AM	3/8/2022 4:59:59 PM	3/18/2022 4:59:59 PM
03/12/2022 10:06:54 AM	3/9/2022 4:59:59 PM	3/19/2022 4:59:59 PM
03/13/2022 10:44:42 AM	3/10/2022 4:59:59 PM	3/20/2022 4:59:59 PM
03/14/2022 11:33:30 AM	3/11/2022 4:59:59 PM	3/21/2022 4:59:59 PM
03/15/2022 12:58:14 PM	3/12/2022 4:59:59 PM	3/22/2022 4:59:59 PM
03/16/2022 4:23:12 PM	3/13/2022 4:59:59 PM	3/23/2022 4:59:59 PM
03/17/2022 6:42:15 PM	3/14/2022 4:59:59 PM	3/24/2022 4:59:59 PM
03/18/2022 7:41:16 PM	3/15/2022 4:59:59 PM	3/25/2022 4:59:59 PM
03/19/2022 8:14:15 PM	3/16/2022 4:59:59 PM	3/26/2022 4:59:59 PM
03/20/2022 9:23:51 PM	3/17/2022 4:59:59 PM	3/27/2022 4:59:59 PM
03/21/2022 10:04:41 PM	3/18/2022 4:59:59 PM	3/28/2022 4:59:59 PM

Die Datumsangaben sind die gleichen wie in Beispiel 2, weisen aber jetzt einen Zeitstempel mit der letzten Millisekunde vor 5 Uhr nachmittags auf, weil der Wert des dritten Arguments, `day_start`, der in der Funktion `dayend()` übergeben wurde, 17/24 war.

Beispiel 4 – Diagrammbeispiel

Ladeskript und Diagrammformel

Überblick

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Der Datensatz und das Szenario in diesem Beispiel sind die Gleichen wie in vorherigen zwei Beispielen. Ihr Unternehmen möchte mit einer Richtlinie arbeiten, bei der der Arbeitstag um 5 Uhr nachmittags beginnt und um 5 Uhr nachmittags am Folgetag endet.

Wie im vorherigen Beispiel erstellen Sie zwei neue Felder:

- `deposit_due_date`: Das Datum, an dem die Anzahlung eingehen sollte. Dies ist das Tagesende drei Tage vor dem `service_date`.
- `final_payment_due_date`: Das Datum, an dem die abschließende Zahlung eingehen sollte. Dies ist das Tagesende sieben Tage nach dem `service_date`.

Ladeskript

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Services:
Load
service_id,
service_date,
amount
Inline
[
service_id, service_date,amount
1,03/11/2022 9:25:14 AM,231.24
2,03/12/2022 10:06:54 AM,567.28
3,03/13/2022 10:44:42 AM,364.28
4,03/14/2022 11:33:30 AM,575.76
5,03/15/2022 12:58:14 PM,638.68
6,03/16/2022 4:23:12 PM,785.38
7,03/17/2022 6:42:15 PM,967.46
8,03/18/2022 7:41:16 PM,287.67
9,03/19/2022 8:14:15 PM,764.45
10,03/20/2022 9:23:51 PM,875.43
11,03/21/2022 10:04:41 PM,957.35
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu:

`service_date`.

Um das Feld `deposit_due_date` zu erstellen, erstellen Sie diese Kennzahl.

```
=dayend(service_date,-3,17/24).
```

Um das Feld `final_payment_due_date` zu erstellen, erstellen Sie dann diese Kennzahl:

```
=dayend(service_date,7,17/24).
```

Ergebnistabelle

service_date	=dayend(service_date,-3,17/24)	=dayend(service_date,7,17/24
03/11/2022	3/8/2022 16:59:59 PM	3/18/2022 16:59:59 PM
03/12/2022	3/9/2022 16:59:59 PM	3/19/2022 16:59:59 PM
03/13/2022	3/10/2022 16:59:59 PM	3/20/2022 16:59:59 PM
03/14/2022	3/11/2022 16:59:59 PM	3/21/2022 16:59:59 PM
03/15/2022	3/12/2022 16:59:59 PM	3/22/2022 16:59:59 PM
03/16/2022	3/13/2022 16:59:59 PM	3/23/2022 16:59:59 PM
03/17/2022	3/14/2022 16:59:59 PM	3/24/2022 16:59:59 PM
03/18/2022	3/15/2022 16:59:59 PM	3/25/2022 16:59:59 PM
03/19/2022	3/16/2022 16:59:59 PM	3/26/2022 16:59:59 PM
03/20/2022	3/17/2022 16:59:59 PM	3/27/2022 16:59:59 PM
03/21/2022	3/18/2022 16:59:59 PM	3/28/2022 16:59:59 PM

Die Werte der neuen Felder sind im `TimestampFormat M/D/YYYY h:mm:ss[.fff] TT`. Da die Funktion `dayend ()` verwendet wurde, sind die Zeitstempelwerte alle die letzte Millisekunde des Tages.

Die Werte für die Fälligkeit der Zahlung sind drei Tage vor dem Fälligkeitsdatum des Dienstes, da das zweite Argument, das in der Funktion `dayend()` übergeben wird, negativ ist.

Die Werte für die Fälligkeit der abschließenden Zahlung sind sieben Tage nach dem Fälligkeitsdatum des Dienstes, da das zweite Argument, das in der Funktion `dayend()` übergeben wird, positiv ist.

Die Datumsangaben weisen einen Zeitstempel mit der letzten Millisekunde vor 17 Uhr auf, weil der Wert des dritten Arguments, `day_start`, der in der Funktion `dayend()` übergeben wurde, 17/24 war.

Argumente

Argument	Beschreibung
time	Der zu berechnende Zeitstempel.
period_no	**period_no** ist eine ganze Zahl oder eine Formel, die eine ganze Zahl ergibt, wobei 0 für den Tag steht, der **time** enthält. Negative Werte von **period_no** stehen für vorangehende, positive Werte für nachfolgende Tage.
day_start	Um anzugeben, dass Tage nicht um Mitternacht beginnen, geben Sie einen Versatz als Bruchteil eines Tages in **day_start** an. So steht beispielsweise 0,125 für 3:00 Uhr.

daylightsaving

Liefert die aktuelle Einstellung bezüglich der Sommerzeit, so wie sie in Windows definiert ist.

Syntax:

```
DaylightSaving( )
```

Rückgabe Datentyp: dual

Beispiel:

```
daylightsaving( )
```

dayname

Diese Funktion liefert den Zeitstempel der ersten Millisekunde des Tages, in dem **time** liegt. Das Ergebnis wird als Datum formatiert.

Syntax:

```
DayName(time[, period_no [, day_start]])
```

Rückgabe Datentyp: dual

Argumente:

Argumente

Argument	Beschreibung
time	Der zu berechnende Zeitstempel.
period_no	**period_no** ist eine ganze Zahl oder eine Formel, die eine ganze Zahl ergibt, wobei 0 für den Tag steht, der **time** enthält. Negative Werte von **period_no** stehen für vorangehende, positive Werte für nachfolgende Tage.
day_start	Um anzugeben, dass Tage nicht um Mitternacht beginnen, geben Sie einen Versatz als Bruchteil eines Tages in **day_start** an. So steht beispielsweise 0,125 für 3:00 Uhr.

Beispiele und Ergebnisse:
In diesen Beispielen wird das Datumsformat **DD/MM/YYYY** verwendet. Das Datumsformat wird im Befehl **SET DateFormat** oben in Ihrem Datenladeskript angegeben. Ändern Sie das Format in den Beispielen nach Bedarf.

Skriptbeispiele

Beispiel	Ergebnis
`dayname('25/01/2013 16:45:00')`	Gibt 25/01/2013 zurück.
`dayname('25/01/2013 16:45:00', -1)`	Gibt 24/01/2013 zurück.
`dayname('25/01/2013 16:45:00', 0, 0.5 )`	Liefert 25/01/2013. Durch die Anzeige des vollständigen Zeitstempels wird der zugrunde liegende numerische Wert angezeigt, der Folgendem entspricht: 25/01/2013 12:00:00.000.

Beispiel:

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Fügen Sie einem Arbeitsblatt in Ihrer App dann die Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um das Ergebnis anzuzeigen.

In diesem Beispiel wird der Wochentag aus einem Zeitstempel ermittelt, der den Anfang des Tages nach jedem Rechnungsdatum in der Tabelle markiert.

```
TempTable:

LOAD RecNo() as InvID, * Inline [

InvDate

28/03/2012

10/12/2012

5/2/2013

31/3/2013

19/5/2013

15/9/2013

11/12/2013

2/3/2014

14/5/2014

13/6/2014

7/7/2014

4/8/2014

];



InvoiceData:

LOAD *,

DayName(InvDate, 1) AS DName

Resident TempTable;
Drop table TempTable;
```

Die sich daraus ergebende Tabelle enthält die ursprünglichen Daten sowie eine Spalte mit dem Rückgabewert der Funktion dayname(). Sie können den vollständigen Zeitstempel anzeigen, indem Sie die Formatierung im Eigenschaftsfenster angeben.

Ergebnistabelle

InvDate	DName
28/03/2012	29/03/2012 00:00:00
10/12/2012	11/12/2012 00:00:00
5/2/2013	07/02/2013 00:00:00
31/3/2013	01/04/2013 00:00:00
19/5/2013	20/05/2013 00:00:00
15/9/2013	16/09/2013 00:00:00
11/12/2013	12/12/2013 00:00:00
2/3/2014	03/03/2014 00:00:00
14/5/2014	15/05/2014 00:00:00
13/6/2014	14/06/2014 00:00:00
7/7/2014	08/07/2014 00:00:00
4/8/2014	05/08/2014 00:00:00

daynumberofquarter

Diese Funktion berechnet die Nummer des Tages des Quartals, in dem der Zeitstempel liegt. Diese Funktion wird verwendet, wenn Sie einen Master-Kalender erstellen.

Syntax:

```
DayNumberOfQuarter(timestamp[,start_month])
```

Rückgabe Datentyp: ganze Zahl

Argumente

Argument	Beschreibung
timestamp	Datum oder Zeitstempel für die Evaluierung.
start_month	Durch Angabe von **start_month** zwischen 2 und 12 (bei fehlender Angabe wird 1 ausgewählt) können Sie den Beginn des Jahres variieren. Beginnt Ihr Geschäftsjahr beispielsweise am 1. März, geben Sie für den Parameter **start_month** = 3 ein.

In diesen Beispielen wird das Datumsformat **DD/MM/YYYY** verwendet. Das Datumsformat wird im Befehl **SET DateFormat** oben in Ihrem Datenladeskript angegeben. Ändern Sie das Format in den Beispielen nach Bedarf.

Funktionsbeispiele

Beispiel	Ergebnis
`DayNumberOfQuarter('12/09/2014')`	Liefert 74, die Nummer des Tages im aktuellen Quartal.
`DayNumberOfQuarter ('12/09/2014',3)`	Liefert 12, die Nummer des Tages im aktuellen Quartal. In diesem Fall beginnt das erste Quartal im März (weil start_ month mit 3 festgelegt wurde). Das bedeutet, dass es sich beim aktuellen Quartal um das dritte Quartal handelt, das am 1. September begonnen hat.

Beispiel 1 – Jahresanfang im Januar (Skript)

Ladeskript und Ergebnisse

Überblick

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein einfacher Datensatz enthält eine Liste der Datumsangaben, die in eine Tabelle mit dem Namen `Calendar` geladen werden. Die `DateFormat`-Standardsystemvariable MM/TT/JJJJ wird verwendet.
- Einen vorangehenden load-Befehl, mit dem ein zusätzliches Feld mit dem Namen `DayNrQtr` erstellt wird, unter Verwendung der Funktion `DayNumberOfQuarter()`.

Abgesehen vom Datum werden der Funktion keine weiteren Parameter bereitgestellt.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Calendar:
Load
        date,
        DayNumberOfQuarter(date) as DayNrQtr
        ;
Load
date
Inline
[
date
01/01/2022
01/10/2022
01/31/2022
02/01/2022
02/10/2022
02/28/2022
03/01/2022
03/31/2022
```

```
04/01/2022
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `daynrqtr`

Ergebnistabelle

date	**daynrqtr**
01/01/2022	1
01/10/2022	10
01/31/2022	31
02/01/2022	32
02/10/2022	41
02/28/2022	59
03/01/2022	61
03/31/2022	91
04/01/2022	1

Der erste Tag des Jahres ist der 1. Januar, weil kein zweites Argument an die Funktion `DayNumberOfQuarter()` übergeben wurde.

Der 1. Januar ist der 1. Tag des Quartals, während der 1. Februar der 32. Tag des Quartals ist. Der 31. März ist der 91. und letzte Tag des Quartals, während der 1. April der 1. Tag des 2. Quartals ist.

Beispiel 2 – Jahresanfang im Februar (Skript)

Ladeskript und Ergebnisse

Überblick

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Den gleichen Datensatz wie im ersten Beispiel.
- Die `DateFormat`-Standardsystemvariable MM/TT/JJJJ wird verwendet.
- Ein Argument `start_month`, das am 1. Februar beginnt. Damit wird der Beginn des Geschäftsjahres auf den 1. Februar festgelegt.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Calendar:
Load
       date,
       DayNumberOfQuarter(date,2) as DayNrQtr
       ;
Load
date
Inline
[
date
01/01/2022
01/10/2022
01/31/2022
02/01/2022
02/10/2022
02/28/2022
03/01/2022
03/31/2022
04/01/2022
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `daynrqtr`

Ergebnistabelle

date	daynrqtr
01/01/2022	62
01/10/2022	71
01/31/2022	92
02/01/2022	1
02/10/2022	10
02/28/2022	28
03/01/2022	30
03/31/2022	60
04/01/2022	61

Der erste Tag des Jahres ist der 1. Februar, weil das zweite Argument, das an die Funktion `DayNumberOfQuarter()` übergeben wurde, 2 war.

Das erste Quartal des Jahres läuft von Februar bis April, während das vierte Quartal von November bis Januar läuft. Dies wird in der Ergebnistabelle gezeigt, wo der 1. Februar der 1. Tag des Quartals ist, während der 31. Januar der 92. und letzte Tag des Quartals ist.

Beispiel 3 – Jahresanfang im Januar (Diagramm)

Ladeskript und Diagrammformel

Überblick

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Den gleichen Datensatz wie im ersten Beispiel.
- Die `DateFormat`-Standardsystemvariable MM/TT/JJJJ wird verwendet.

In diesem Beispiel wird jedoch der unveränderte Datensatz in die Anwendung geladen. Der Wert des Tages des Quartals wird anhand einer Kennzahl in einem Diagrammobjekt berechnet.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Calendar:
Load
date
Inline
[
date
01/01/2022
01/10/2022
01/31/2022
02/01/2022
02/10/2022
02/28/2022
03/01/2022
03/31/2022
04/01/2022
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu: `date`.

Erstellen Sie die folgende Kennzahl:

```
=daynumberofquarter(date)
```

Ergebnistabelle

date	=daynumberofquarter(date)
01/01/2022	1
01/10/2022	10
01/31/2022	31
02/01/2022	32
02/10/2022	41
02/28/2022	59
03/01/2022	61
03/31/2022	91
04/01/2022	1

Der erste Tag des Jahres ist der 1. Januar, weil kein zweites Argument an die Funktion `DayNumberOfQuarter()` übergeben wurde.

Der 1. Januar ist der 1. Tag des Quartals, während der 1. Februar der 32. Tag des Quartals ist. Der 31. März ist der 91. und letzte Tag des Quartals, während der 1. April der 1. Tag des 2. Quartals ist.

Beispiel 4 – Jahresanfang im Februar (Diagramm)

Ladeskript und Diagrammformel

Überblick

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Den gleichen Datensatz wie im ersten Beispiel.
- Die `DateFormat`-Standardsystemvariable MM/TT/JJJJ wird verwendet.
- Das Geschäftsjahr läuft vom 1. Februar bis zum 31. Januar.

In diesem Beispiel wird jedoch der unveränderte Datensatz in die Anwendung geladen. Der Wert des Tages des Quartals wird anhand einer Kennzahl in einem Diagrammobjekt berechnet.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Calendar:
Load
date
Inline
[
date
```

```
01/01/2022
01/10/2022
01/31/2022
02/01/2022
02/10/2022
02/28/2022
03/01/2022
03/31/2022
04/01/2022
];
```

Diagrammobjekt

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu: `date`.

Erstellen Sie die folgende Kennzahl:

```
=daynumberofquarter(date,2)
```

Ergebnisse

Ergebnistabelle

date	**=daynumberofquarter(date,2)**
01/01/2022	62
01/10/2022	71
01/31/2022	92
02/01/2022	1
02/10/2022	10
02/28/2022	28
03/01/2022	30
03/31/2022	60
04/01/2022	61

Der erste Tag des Jahres ist der 1. Februar, weil das zweite Argument, das an die Funktion `DayNumberOfQuarter()` übergeben wurde, 2 war.

Das erste Quartal des Jahres läuft von Februar bis April, während das vierte Quartal von November bis Januar läuft. Dies wird in der Ergebnistabelle gezeigt, wo der 1. Februar ist der 1. Tag des Quartals ist, während der 31. Januar der 92. und letzte Tag des Quartals ist.

daynumberofyear

Diese Funktion berechnet die Nummer des Tages des Jahres, in dem der Zeitstempel liegt. Die Berechnung erfolgt ab der ersten Millisekunde des ersten Tags des Jahres, aber der Beginn des ersten Monats kann festgelegt werden.

Syntax:

```
DayNumberOfYear(timestamp[,start_month])
```

Rückgabe Datentyp: ganze Zahl

Argumente

Argument	Beschreibung
timestamp	Datum oder Zeitstempel für die Evaluierung.
start_month	Durch Angabe von **start_month** zwischen 2 und 12 (bei fehlender Angabe wird 1 ausgewählt) können Sie den Beginn des Jahres variieren. Beginnt Ihr Geschäftsjahr beispielsweise am 1. März, geben Sie für den Parameter **start_month** = 3 ein.

In diesen Beispielen wird das Datumsformat **DD/MM/YYYY** verwendet. Das Datumsformat wird im Befehl **SET DateFormat** oben in Ihrem Datenladeskript angegeben. Ändern Sie das Format in den Beispielen nach Bedarf.

Funktionsbeispiele

Beispiel	Ergebnis
`DayNumberOfYear( '12/09/2014' )`	Liefert 256, die Nummer des Tages vom ersten Tag des Jahres an gezählt.
`DayNumberOfYear( '12/09/2014',3 )`	Liefert 196, die Nummer des Tages vom 1. März an gezählt.

Beispiel 1 – Jahresanfang im Januar (Skript)

Ladeskript und Ergebnisse

Überblick

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein einfacher Datensatz enthält eine Liste der Datumsangaben, die in eine Tabelle mit dem Namen `Calendar` geladen werden. Die `DateFormat`-Standardsystemvariable MM/TT/JJJJ wird verwendet.
- Einen vorangehenden load-Befehl, mit dem ein zusätzliches Feld mit dem Namen `daynryear` erstellt wird, unter Verwendung der Funktion `DayNumberOfYear()`.

Abgesehen vom Datum werden der Funktion keine weiteren Parameter bereitgestellt.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Calendar:
Load
       date,
       DayNumberOfYear(date) as daynryear
       ;
Load
date
Inline
[
date
01/01/2022
01/10/2022
01/31/2022
02/01/2022
02/10/2022
06/30/2022
07/26/2022
10/31/2022
11/01/2022
12/31/2022
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `daynryear`

Ergebnistabelle

date	daynryear
01/01/2022	1
01/10/2022	10
01/31/2022	31
02/01/2022	32
02/10/2022	41
06/30/2022	182
07/26/2022	208
10/31/2022	305
11/01/2022	306
12/31/2022	366

Der erste Tag des Jahres ist der 1. Januar, weil kein zweites Argument an die Funktion `DayNumberOfYear()` übergeben wurde.

Der 1. Januar ist der 1. Tag des Quartals, während der 1. Februar der 32. Tag des Jahres ist. Der 30. Juni ist der 182. Tag, während der 31. Dezember der 366. und letzte Tag des Jahres ist.

Beispiel 2 – Jahresanfang im November (Skript)

Ladeskript und Ergebnisse

Überblick

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Den gleichen Datensatz wie im ersten Beispiel.
- Die `DateFormat`-Standardsystemvariable MM/TT/JJJJ wird verwendet.
- Ein Argument `start_month`, das am 1. November beginnt. Damit wird der Beginn des Geschäftsjahres auf den 1. November festgelegt.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Calendar:
Load
       date,
       DayNumberOfYear(date,11) as daynryear
       ;
Load
date
Inline
[
date
01/01/2022
01/10/2022
01/31/2022
02/01/2022
02/10/2022
06/30/2022
07/26/2022
10/31/2022
11/01/2022
12/31/2022
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- date
- daynryear

Ergebnistabelle

date	daynryear
01/01/2022	62
01/10/2022	71
01/31/2022	92
02/01/2022	93
02/10/2022	102
06/30/2022	243
07/26/2022	269
10/31/2022	366
11/01/2022	1
12/31/2022	61

Der erste Tag des Jahres ist der 1. November, weil das zweite Argument, das an die Funktion `DayNumberOfYear()` übergeben wurde, 11 war.

Der 1. Januar ist der 1. Tag des Quartals, während der 1. Februar der 32. Tag des Jahres ist. Der 30. Juni ist der 182. Tag, während der 31. Dezember der 366. und letzte Tag des Jahres ist.

Beispiel 3 – Jahresanfang im Januar (Diagramm)

Ladeskript und Diagrammformel

Überblick

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Den gleichen Datensatz wie im ersten Beispiel.
- Die `DateFormat`-Standardsystemvariable MM/TT/JJJJ wird verwendet.

In diesem Beispiel wird jedoch der unveränderte Datensatz in die Anwendung geladen. Der Wert des Tages des Quartals wird anhand einer Kennzahl in einem Diagrammobjekt berechnet.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Calendar:
Load
```

```
date
Inline
[
date
01/01/2022
01/10/2022
01/31/2022
02/01/2022
02/10/2022
06/30/2022
07/26/2022
10/31/2022
11/01/2022
12/31/2022
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu: `date`.

Erstellen Sie die folgende Kennzahl:

```
=daynumberofyear(date)
```

Ergebnistabelle

date	**=daynumberofyear(date)**
01/01/2022	1
01/10/2022	10
01/31/2022	31
02/01/2022	32
02/10/2022	41
06/30/2022	182
07/26/2022	208
10/31/2022	305
11/01/2022	306
12/31/2022	366

Der erste Tag des Jahres ist der 1. Januar, weil kein zweites Argument an die Funktion `DayNumberOfYear()` übergeben wurde.

Der 1. Januar ist der 1. Tag des Jahres , während der 1. Februar der 32. Tag des Jahres ist. Der 30. Juni ist der 182. Tag, während der 31. Dezember der 366. und letzte Tag des Jahres ist.

Beispiel 4 – Jahresanfang im November (Diagramm)

Ladeskript und Diagrammformel

Überblick

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Den gleichen Datensatz wie im ersten Beispiel.
- Die `DateFormat`-Standardsystemvariable MM/TT/JJJJ wird verwendet.
- Das Geschäftsjahr läuft vom 1. November bis zum 31. Oktober.

In diesem Beispiel wird jedoch der unveränderte Datensatz in die Anwendung geladen. Der Wert des Tages des Jahres wird anhand einer Kennzahl in einem Diagrammobjekt berechnet.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';
Calendar:
Load
date
Inline
[
date
01/01/2022
01/10/2022
01/31/2022
02/01/2022
02/10/2022
06/30/2022
07/26/2022
10/31/2022
11/01/2022
12/31/2022
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu: `date`.

Erstellen Sie die folgende Kennzahl:

```
=daynumberofyear(date)
```

Ergebnistabelle

date	**=daynumberofyear(date,11)**
01/01/2022	62

date	=daynumberofyear(date,11)
01/10/2022	71
01/31/2022	92
02/01/2022	93
02/10/2022	102
06/30/2022	243
07/26/2022	269
10/31/2022	366
11/01/2022	1
12/31/2022	61

Der erste Tag des Jahres ist der 1. November, weil das zweite Argument, das an die Funktion `DayNumberOfYear()` übergeben wurde, 11 war.

Das Geschäftsjahr läuft von November bis Oktober. Dies wird in der Ergebnistabelle gezeigt, wo der 1. November der 1. Tag des Jahres ist, während der 31. Oktober der 366. und letzte Tag des Jahres ist.

daystart

Diese Funktion liefert den Zeitstempel der ersten Millisekunde des Tages, in dem das Argument **time** liegt. Das Ergebnis wird entsprechend dem im Skript definierten **TimestampFormat** formatiert.

Syntax:

```
DayStart(time[, [period_no[, day_start]])
```

Rückgabe Datentyp: dual

Argumente

Argument	Beschreibung
time	Der zu berechnende Zeitstempel.
period_no	**period_no** ist eine ganze Zahl oder eine Formel, die eine ganze Zahl ergibt, wobei 0 für den Tag steht, der **time** enthält. Negative Werte von **period_no** stehen für vorangehende, positive Werte für nachfolgende Tage.
day_start	Um anzugeben, dass Tage nicht um Mitternacht beginnen, geben Sie einen Versatz als Bruchteil eines Tages in **day_start** an. So steht beispielsweise 0,125 für 3:00 Uhr. Um den Versatz zu erstellen, teilen Sie also die Startuhrzeit durch 24 Stunden. Damit beispielsweise ein Tag um 7:00 Uhr beginnt, verwenden Sie den Bruch 7/24.

Verwendung

Die Funktion `daystart()` wird in der Regel als Teil einer Formel verwendet, wenn in der Berechnung der Teil des Tages verwendet werden soll, der bereits verstrichen ist. Beispielsweise kann sie verwendet werden, um den Gesamtlohn zu berechnen, der bisher von den Mitarbeitern am Tag verdient wurde.

In diesen Beispielen wird das Zeitstempelformat `'M/D/YYYY h:mm:ss[.fff] TT'` verwendet. Das Zeitstempelformat wird in der Anweisung `SET TimeStamp` oben in Ihrem Datenladeskript angegeben. Ändern Sie das Format in den Beispielen nach Bedarf.

Funktionsbeispiele

Beispiel	**Ergebnis**
`daystart('01/25/2013 4:45:00 PM')`	Gibt 1/25/2013 12:00:00 AM zurück.
`daystart('1/25/2013 4:45:00 PM', -1)`	Gibt 1/24/2013 12:00:00 AM zurück.
`daystart('1/25/2013 16:45:00',0,0.5 )`	Gibt 1/25/2013 12:00:00 PM zurück.

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Beispiel 1 – Einfaches Beispiel

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein einfacher Datensatz enthält eine Liste der Datumsangaben, die in eine Tabelle mit dem Namen `Calendar` geladen werden.
- Die Standardsystemvariable `TimeStampFormat` (`M/D/YYYY h:mm:ss[.fff] TT`) wird verwendet.

- Mit einem vorangehenden load-Befehl wird unter Verwendung der Funktion `daystart()` ein zusätzliches Feld mit dem Namen `SOD_timestamp` erstellt.

Abgesehen vom Datum werden der Funktion keine weiteren Parameter bereitgestellt.

Ladeskript

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Calendar:
    Load
        date,
        daystart(date) as SOD_timestamp
        ;
Load
date
Inline
[
date
03/11/2022 1:47:15 AM
03/12/2022 4:34:58 AM
03/13/2022 5:15:55 AM
03/14/2022 9:25:14 AM
03/15/2022 10:06:54 AM
03/16/2022 10:44:42 AM
03/17/2022 11:33:30 AM
03/18/2022 12:58:14 PM
03/19/2022 4:23:12 PM
03/20/2022 6:42:15 PM
03/21/2022 7:41:16 PM
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `SOD_timestamp`

Ergebnistabelle

date	SOD_timestamp
03/11/2022 1:47:15 AM	3/11/2022 12:00:00 AM
03/12/2022 4:34:58 AM	3/12/2022 12:00:00 AM
03/13/2022 5:15:55 AM	3/13/2022 12:00:00 AM
03/14/2022 9:25:14 AM	3/14/2022 12:00:00 AM

date	SOD_timestamp
03/15/2022 10:06:54 AM	3/15/2022 12:00:00 AM
03/16/2022 10:44:42 AM	3/16/2022 12:00:00 AM
03/17/2022 11:33:30 AM	3/17/2022 12:00:00 AM
03/18/2022 12:58:14 PM	3/18/2022 12:00:00 AM
03/19/2022 4:23:12 PM	3/19/2022 12:00:00 AM
03/20/2022 6:42:15 PM	3/20/2022 12:00:00 AM
03/21/2022 7:41:16 PM	3/21/2022 12:00:00 AM

Wie in der Tabelle oben zu sehen ist, wird der Tagesende-Zeitstempel für jedes Datum in unserem Datensatz erstellt. Der Zeitstempel hat das Format der Systemvariablen `TimestampFormat M/D/YYYY h:mm:ss[.fff] TT`.

Beispiel 2 – period_no

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz, der Strafzettel für Falschparken enthält, wird in eine Tabelle mit dem Namen `Fines` geladen. Der Datensatz umfasst die folgenden Felder:
  - `id`
  - `due_date`
  - `number_plate`
  - `amount`
- Eine vorangehende load-Anweisung, die die Funktion `daystart()` verwendet und alle drei Parameter bereitstellt: `time`, `period_no` und `day_start`. Diese vorangehende load-Anweisung erstellt die folgenden neuen Datumsfelder:
  - Ein Datumsfeld `early_repayment_period`, das sieben Tage vor Fälligkeit der Zahlung beginnt.
  - Ein Datumsfeld `late_penalty_period`, das 14 Tage nach Fälligkeit der Zahlung beginnt.

Ladeskript

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Fines:
    Load
        *,
        daystart(due_date,-7) as early_repayment_period,
        daystart(due_date,14) as late_penalty_period
        ;
```

```
Load
*
Inline
[
id, due_date, number_plate,amount
1,02/11/2022, 573RJG,50.00
2,03/25/2022, SC41854,50.00
3,04/14/2022, 8EHZ378,50.00
4,06/28/2022, 8HSS198,50.00
5,08/15/2022, 1221665,50.00
6,11/16/2022, EAK473,50.00
7,01/17/2023, KD6822,50.00
8,03/22/2023, 1GGLB,50.00
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- due_date
- early_repayment_period
- late_penalty_period

Ergebnistabelle

due_date	early_repayment_period	late_penalty_period
02/11/2022 9:25:14 AM	2/4/2022 12:00:00 AM	2/25/2022 12:00:00 AM
03/25/2022 10:06:54 AM	3/18/2022 12:00:00 AM	4/8/2022 12:00:00 AM
04/14/2022 10:44:42 AM	4/7/2022 12:00:00 AM	4/28/2022 12:00:00 AM
06/28/2022 11:33:30 AM	6/21/2022 12:00:00 AM	7/12/2022 12:00:00 AM
08/15/2022 12:58:14 PM	8/8/2022 12:00:00 AM	8/29/2022 12:00:00 AM
11/16/2022 4:23:12 PM	11/9/2022 12:00:00 AM	11/30/2022 12:00:00 AM
01/17/2023 6:42:15 PM	1/10/2023 12:00:00 AM	1/31/2023 12:00:00 AM
03/22/2023 7:41:16 PM	3/15/2023 12:00:00 AM	4/5/2023 12:00:00 AM

Die Werte der neuen Felder sind im `TimestampFormat M/DD/YYYY tt`. Da die Funktion `daystart()` verwendet wurde, sind die Zeitstempelwerte alle die erste Millisekunde des Tages.

Die Werte für den frühen Zahlungszeitraum liegen sieben Tage vor dem Fälligkeitsdatum, da das zweite Argument, das in der Funktion `daystart()` übergeben wird, negativ ist.

Die Werte für den späten Zahlungszeitraum liegen 14 Tage nach dem Fälligkeitsdatum, da das zweite Argument, das in der Funktion `daystart()` übergeben wird, positiv ist.

Beispiel 3 – day_start

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Derselbe Datensatz und dasselbe Szenario wie im vorigen Beispiel.
- Die gleiche vorangehende load-Anweisung wie im vorigen Beispiel.

In diesem Beispiel wird der Beginn und das Ende des Arbeitstags täglich auf 7:00 AM festgelegt.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Fines:
    Load
        *,
        daystart(due_date,-7,7/24) as early_repayment_period,
        daystart(due_date,14, 7/24) as late_penalty_period
        ;
Load
*
Inline
[
id, due_date, number_plate,amount
1,02/11/2022, 573RJG,50.00
2,03/25/2022, SC41854,50.00
3,04/14/2022, 8EHZ378,50.00
4,06/28/2022, 8HSS198,50.00
5,08/15/2022, 1221665,50.00
6,11/16/2022, EAK473,50.00
7,01/17/2023, KD6822,50.00
8,03/22/2023, 1GGLB,50.00
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- due_date
- early_repayment_period
- late_penalty_period

Ergebnistabelle

due_date	early_repayment_period	late_penalty_period
02/11/2022	2/3/2022 7:00:00 AM	2/24/2022 7:00:00 AM
03/25/2022	3/17/2022 7:00:00 AM	4/7/2022 7:00:00 AM
04/14/2022	4/6/2022 7:00:00 AM	4/27/2022 7:00:00 AM
06/28/2022	6/20/2022 7:00:00 AM	7/11/2022 7:00:00 AM
08/15/2022	8/7/2022 7:00:00 AM	8/28/2022 7:00:00 AM
11/16/2022	11/8/2022 7:00:00 AM	11/29/2022 7:00:00 AM
01/17/2023	1/9/2023 7:00:00 AM	1/30/2023 7:00:00 AM
03/22/2023	3/14/2023 7:00:00 AM	4/4/2023 7:00:00 AM

Die Datumswerte weisen jetzt einen Zeitstempel von 7:00 AM auf, weil der Wert des Arguments `day_start`, das in die Funktion `daystart()` übergeben wurde, 7/24 war. Damit wird der Tagesbeginn auf 7:00 AM festgelegt.

Da das Feld `due_date` keinen Zeitstempel hat, wird es als 12:00 AM behandelt, gehört also noch zum vorigen Tag, da der Tag um 7:00 AM beginnt und endet. Daher beginnt der frühe Zahlungszeitraum für eine Geldstrafe, die am 11. Februar fällig ist, am 3. Februar um 7:00 Uhr.

Beispiel 4 – Diagrammobjektbeispiel

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

In diesem Beispiel wird derselbe Datensatz und dasselbe Szenario wie im vorigen Beispiel verwendet.

Es wird aber nur die ursprüngliche Tabelle `Fines` in die Anwendung geladen, zusammen mit den beiden zusätzlichen Fälligkeitsdatumswerten, die in einem Diagrammobjekt berechnet werden.

Ladeskript

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Fines:
    Load
*
Inline
[
id, due_date, numer_plate,amount
1,02/11/2022 9:25:14 AM, 573RJG,50.00
2,03/25/2022 10:06:54 AM, SC41854,50.00
3,04/14/2022 10:44:42 AM, 8EHZ378,50.00
4,06/28/2022 11:33:30 AM, 8HSS198,50.00
5,08/15/2022 12:58:14 PM, 1221665,50.00
```

```
6,11/16/2022 4:23:12 PM, EAK473,50.00
7,01/17/2023 6:42:15 PM, KD6822,50.00
8,03/22/2023 7:41:16 PM, 1GGLB,50.00
];
```

Ergebnisse

Gehen Sie folgendermaßen vor:

1. Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu: `due_date`.
2. Um das Feld `early_repayment_period` zu erstellen, erstellen Sie die folgende Kennzahl: `=daystart(due_date,-7,7/24)`
3. Um das Feld `late_penalty_period` zu erstellen, erstellen Sie die folgende Kennzahl: =daystart(due_date,14,7/24)

Ergebnistabelle

due_date	=daystart(due_date,-7,7/24)	=daystart(due_date,14,7/24)
02/11/2022 9:25:14 AM	2/4/2022 7:00:00 AM	2/25/2022 7:00:00 AM
03/25/2022 10:06:54 AM	3/18/2022 7:00:00 AM	4/8/2022 7:00:00 AM
04/14/2022 10:44:42 AM	4/7/2022 7:00:00 AM	4/28/2022 7:00:00 AM
06/28/2022 11:33:30 AM	6/21/2022 7:00:00 AM	7/12/2022 7:00:00 AM
08/15/2022 12:58:14 PM	8/8/2022 7:00:00 AM	8/29/2022 7:00:00 AM
11/16/2022 4:23:12 PM	11/9/2022 7:00:00 AM	11/30/2022 7:00:00 AM
01/17/2023 6:42:15 PM	1/10/2023 7:00:00 AM	1/31/2023 7:00:00 AM
03/22/2023 7:41:16 PM	3/15/2023 7:00:00 AM	4/5/2023 7:00:00 AM

Die Werte der neuen Felder sind im `TimestampFormat M/D/YYYY h:mm:ss[.fff] TT`. Da die Funktion `daystart()` verwendet wurde, entsprechen die Zeitstempelwerte der ersten Millisekunde des Tages.

Die Werte für den frühen Zahlungszeitraum liegen sieben Tage vor dem Fälligkeitsdatum, da das zweite Argument, das in der Funktion `daystart()` übergeben wurde, negativ war.

Die Werte für den späten Zahlungszeitraum liegen 14 Tage nach dem Fälligkeitsdatum, da das zweite Argument, das in der Funktion `daystart()` übergeben wurde, positiv war.

Die Datumswerte weisen einen Zeitstempel von 7:00 AM auf, weil der Wert des dritten Arguments, das an die Funktion `daystart()` (`day_start`) übergeben wurde, 7/24 war.

firstworkdate

Die Funktion **firstworkdate** liefert das späteste Startdatum zur Vollendung einer gewissen Zahl von **no_of_workdays** (Montag bis Freitag) bis zu einem vorgegebenen **end_date**, unter Berücksichtigung eventueller Feiertage, die als weitere Parameter definiert werden können. **end_date** und **holiday** müssen gültige Datumsangaben oder Zeitstempel sein.

Syntax:

```
firstworkdate(end_date, no_of_workdays {, holiday} )
```

Rückgabe Datentyp: ganze Zahl

Argumente:

Argumente

Argument	Beschreibung
end_date	Der Zeitstempel des Enddatums für die Evaluierung.
no_of_ workdays	Die Anzahl der zu erreichenden Arbeitstage.
holiday	Feiertagszeiträume, die von den Arbeitstagen auszuschließen sind. Ein Feiertag wird als ein Datum mit Zeichenfolgenkonstante angegeben. Sie können mehrere Feiertagstermine getrennt durch Kommas festlegen. **Beispiel:** `'12/25/2013', '12/26/2013', '12/31/2013', '01/01/2014'`

Beispiele und Ergebnisse:
In diesen Beispielen wird das Datumsformat **DD/MM/YYYY** verwendet. Das Datumsformat wird im Befehl **SET DateFormat** oben in Ihrem Datenladeskript angegeben. Ändern Sie das Format in den Beispielen nach Bedarf.

Skriptbeispiele

Beispiel	Ergebnis
`firstworkdate ('29/12/2014', 9)`	Liefert '17/12/2014.
`firstworkdate ('29/12/2014', 9, '25/12/2014', '26/12/2014')`	Liefert 15/12/2014, weil ein Feiertagszeitraum von zwei Tagen berücksichtigt wird.

Beispiel:

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Fügen Sie einem Arbeitsblatt in Ihrer App dann die Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um das Ergebnis anzuzeigen.

```
ProjectTable:
LOAD *, recno() as InvID, INLINE [
EndDate
28/03/2015
10/12/2015
5/2/2016
31/3/2016
19/5/2016
15/9/2016
] ;
NrDays:
Load *,
FirstWorkDate(EndDate,120) As StartDate
```

```
Resident ProjectTable;
Drop table ProjectTable;
```

Die entstehende Tabelle enthält die zurückgegebenen Werte von FirstWorkDate für jeden der Datensätze in der Tabelle.

Ergebnistabelle

InvID	EndDate	StartDate
1	28/03/2015	13/10/2014
2	10/12/2015	26/06/2015
3	5/2/2016	24/08/2015
4	31/3/2016	16/10/2015
5	19/5/2016	04/12/2015
6	15/9/2016	01/04/2016

GMT

Diese Funktion liefert die aktuelle Greenwich Mean Time der Regionaleinstellungen. Die Funktion gibt Werte im Systemvariablenformat `TimestampFormat` zurück.

Sooft die App geladen wird, wird jede Ladeskripttabelle, jede Variable oder jedes Diagrammobjekt, die bzw. das die Funktion `GMT` verwendet, an die letzte aktuelle Greenwich Mean Time angepasst, die der Systemuhr entnommen wird.

Syntax:

```
GMT( )
```

Rückgabe Datentyp: dual

In diesen Beispielen wird das Zeitstempelformat `M/D/YYYY h:mm:ss[.fff] TT` verwendet. Das Datumsformat wird im Befehl `SET TimestampFormat` oben in Ihrem Datenladeskript angegeben. Ändern Sie das Format in den Beispielen nach Bedarf.

Funktionsbeispiele

Beispiel	Ergebnis
`GMT()`	3/28/2022 2:47:36 PM

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben.

Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Beispiel 1 – Variable (Skript)

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein. In diesem Beispiel wird die aktuelle Greenwich Mean Time mit der Funktion `GMT` als Variable im Ladeskript festgelegt.

Ladeskript

```
LET vGMT = GMT();
```

Ergebnisse

Laden Sie die Daten und erstellen Sie ein Arbeitsblatt. Erstellen Sie ein Textfeld mithilfe des Diagrammobjekts **Text und Bild**.

Fügen Sie dem Textfeld die folgende Kennzahl hinzu:

```
=vGMT
```

Das Textfeld muss eine Textzeile mit Datum und Uhrzeit ähnlich der folgenden enthalten:

```
3/28/2022 2:47:36 PM
```

Beispiel 2 – Jahresanfang im November (Skript)

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz, der überfällige Bibliotheksbücher enthält, wird in eine Tabelle mit dem Namen `Overdue` geladen. Die `DateFormat`-Standardsystemvariable MM/TT/JJJJ wird verwendet.

- Es wird ein neues Feld mit dem Namen days_overdue erstellt, das berechnet, seit wie vielen Tagen jedes Buch überfällig ist.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Overdue:
    Load
        *,
        Floor(GMT()-due_date) as days_overdue
        ;
Load
*
Inline
[
cust_id,book_id,due_date
1,4,01/01/2021,
2,24,01/10/2021,
6,173,01/31/2021,
31,281,02/01/2021,
86,265,02/10/2021,
52,465,06/30/2021,
26,537,07/26/2021,
92,275,10/31/2021,
27,455,11/01/2021,
27,46,12/31/2021
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- due_date
- book_id
- days_overdue

Ergebnistabelle

due_date	book_id	days_overdue
01/01/2021	4	455
01/10/2021	24	446
01/31/2021	173	425
02/01/2021	281	424
02/10/2021	265	415
06/30/2021	465	275

due_date	book_id	days_overdue
07/26/2021	537	249
10/31/2021	275	152
11/01/2021	455	151
12/31/2021	46	91

Die Werte im Feld `days_overdue` werden berechnet, indem der Unterschied zwischen der aktuellen Greenwich Mean Time mit der Funktion `GMT()` und dem ursprünglichen Fälligkeitsdatum ermittelt wird. Um nur die Tage zu berechnen, werden die Ergebnisse mit der Funktion `Floor()` auf die nächste Ganzzahl abgerundet.

Beispiel 3 – Diagrammobjekt (Diagramm)

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein. Das Ladeskript enthält den gleichen Datensatz wie das vorige Beispiel. Die `DateFormat`-Standardsystemvariable MM/TT/JJJJ wird verwendet.

In diesem Beispiel wird jedoch der unveränderte Datensatz in die Anwendung geladen. Der Wert der Anzahl der überfälligen Tage wird anhand einer Kennzahl in einem Diagrammobjekt berechnet.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Overdue:
Load
*
Inline
[
cust_id,book_id,due_date
1,4,01/01/2021,
2,24,01/10/2021,
6,173,01/31/2021,
31,281,02/01/2021,
86,265,02/10/2021,
52,465,06/30/2021,
26,537,07/26/2021,
92,275,10/31/2021,
27,455,11/01/2021,
27,46,12/31/2021
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- due_date
- book_id

Erstellen Sie die folgende Kennzahl:

```
=Floor(GMT() - due_date)
```

Ergebnistabelle

due_date	book_id	=Floor(GMT()-due_date)
01/01/2021	4	455
01/10/2021	24	446
01/31/2021	173	425
02/01/2021	281	424
02/10/2021	265	415
06/30/2021	465	275
07/26/2021	537	249
10/31/2021	275	152
11/01/2021	455	151
12/31/2021	46	91

Die Werte im Feld days_overdue werden berechnet, indem der Unterschied zwischen der aktuellen Greenwich Mean Time mit der Funktion GMT() und dem ursprünglichen Fälligkeitsdatum ermittelt wird. Um nur die Tage zu berechnen, werden die Ergebnisse mit der Funktion Floor() auf die nächste Ganzzahl abgerundet.

hour

Diese Funktion liefert die Stunde als ganze Zahl, wenn **expression** entsprechend dem Standardformat als Uhrzeit interpretiert wird.

Syntax:

```
hour(expression)
```

Rückgabe Datentyp: ganze Zahl

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Funktionsbeispiele

Beispiel	Ergebnis
`hour( '09:14:36' )`	Der angegebene Textstring wird implizit in einen Zeitstempel konvertiert, da er dem in der Variablen „TimestampFormat“ definierten Zeitstempelformat entspricht. Die Formel gibt 9 zurück.
`hour( '0.5555' )`	Die Formel gibt 13 zurück (da 0.5555 = 13:19:55).

Beispiel 1 – Variable (Skript)

Ladeskript und Ergebnisse

Überblick

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Einen Datensatz, der Transaktionen nach Zeitstempel enthält
- Die `TimeStamp`-Standardsystemvariable `(M/D/YYYY h:mm:ss[.fff] TT)`.

Erstellen Sie ein Feld „`hour`“, das berechnet, wann Käufe stattfanden.

Ladeskript

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Transactions:
  Load
    *,
    hour(date) as hour
```

```
    ;
Load
*
Inline
[
id,date,amount
9497,'2022-01-05 19:04:57',47.25,
9498,'2022-01-03 14:21:53',51.75,
9499,'2022-01-03 05:40:49',73.53,
9500,'2022-01-04 18:49:38',15.35,
9501,'2022-01-01 22:10:22',31.43,
9502,'2022-01-05 19:34:46',13.24,
9503,'2022-01-04 22:58:34',74.34,
9504,'2022-01-06 11:29:38',50.00,
9505,'2022-01-02 08:35:54',36.34,
9506,'2022-01-06 08:49:09',74.23
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `hour`

Ergebnistabelle

date	hour
2022-01-01 22:10:22	22
2022-01-02 08:35:54	8
2022-01-03 05:40:49	5
2022-01-03 14:21:53	14
2022-01-04 18:49:38	18
2022-01-04 22:58:34	22
2022-01-05 19:04:57	19
2022-01-05 19:34:46	19
2022-01-06 08:49:09	8
2022-01-06 11:29:38	11

Die Werte im Feld „hour“ werden anhand der Funktion `hour()` erstellt und übergeben das Datum als die Formel im vorangehenden load-Befehl.

Beispiel 2 – Diagrammobjekt (Diagramm)

Ladeskript und Diagrammformel

Überblick

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Den gleichen Datensatz wie im ersten Beispiel.
- Die `TimeStamp`-Standardsystemvariable `(M/D/YYYY h:mm:ss[.fff] TT)`.

In diesem Beispiel wird jedoch der unveränderte Datensatz in die Anwendung geladen. Die Werte für „`hour`“ werden anhand einer Kennzahl in einem Diagrammobjekt berechnet.

Ladeskript

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Transactions:
Load
*
Inline
[
id,date,amount
9497,'2022-01-05 19:04:57',47.25,
9498,'2022-01-03 14:21:53',51.75,
9499,'2022-01-03 05:40:49',73.53,
9500,'2022-01-04 18:49:38',15.35,
9501,'2022-01-01 22:10:22',31.43,
9502,'2022-01-05 19:34:46',13.24,
9503,'2022-01-04 22:58:34',74.34,
9504,'2022-01-06 11:29:38',50.00,
9505,'2022-01-02 08:35:54',36.34,
9506,'2022-01-06 08:49:09',74.23
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu: `date`.

Erstellen Sie die folgende Kennzahl, um „`hour`“ zu berechnen:

```
=hour(date)
```

Ergebnistabelle

due_date	=hour(date)
2022-01-01 22:10:22	22
2022-01-02 08:35:54	8
2022-01-03 05:40:49	5

due_date	=hour(date)
2022-01-03 14:21:53	14
2022-01-04 18:49:38	18
2022-01-04 22:58:34	22
2022-01-05 19:04:57	19
2022-01-05 19:34:46	19
2022-01-06 08:49:09	8
2022-01-06 11:29:38	11

Die Werte für „hour“ werden erstellt, indem die Funktion `hour()` verwendet und das Datum als Formel in einer Kennzahl für das Diagrammobjekt übergeben wird.

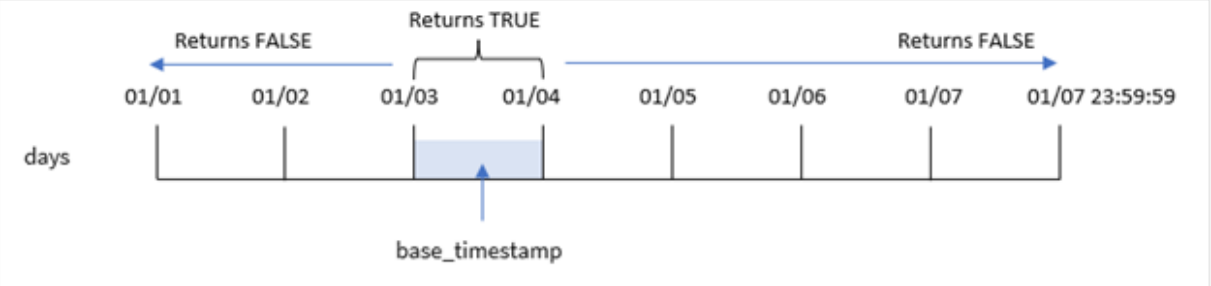
inday

Diese Funktion liefert True, wenn **timestamp** innerhalb des Tages liegt, der **base_timestamp** enthält.

Syntax:

```
InDay (timestamp, base_timestamp, period_no[, day_start])
```

Diagramm der Funktion `inday`



Die Funktion `inday()` verwendet das Argument `base_timestamp`, um zu identifizieren, auf welchen Tag der Zeitstempel fällt. Die Startzeit des Tages ist standardmäßig Mitternacht, aber Sie können die Startzeit des Tages mit dem Argument `day_start` der Funktion `inday()` ändern. Nachdem dieser Tag definiert ist, gibt die Funktion boolesche Ergebnisse zurück, wenn die vorgegebenen Zeitstempelwerte mit diesem Tag verglichen werden.

Verwendung

Die Funktion `inday()` gibt einen booleschen Wert zurück. In der Regel wird dieser Funktionstyp als Bedingung in einer `if expression` verwendet. Dies gibt eine Aggregierung oder Berechnung zurück, abhängig davon, ob ein ausgewertetes Datum auf den Tag des betreffenden Zeitstempels fällt.

Beispielsweise kann die Funktion `inday()` verwendet werden, um alle an einem bestimmten Tag gefertigten Geräte zu identifizieren.

Rückgabe Datentyp: Boolesch

In Qlik Sense wird der boolesche Wert „wahr“ durch -1 dargestellt, der Wert „falsch“ durch 0.

Argumente

Argument	Beschreibung
timestamp	Datum und Uhrzeit, die mit base_timestamp verglichen werden sollen.
base_timestamp	Datum und Uhrzeit, die zur Interpretation des Zeitstempels verwendet werden.
period_no	Mit period_no kann ein anderer Tagesbeginn definiert werden. period_no ist eine ganze Zahl, wobei 0 für den Tag steht, der base_timestamp enthält. Negative Werte von period_no stehen für vorangehende, positive Werte für nachfolgende Tage.
day_start	Wenn Sie mit Tagen arbeiten möchten, die nicht um Mitternacht beginnen, definieren Sie durch Eingabe einer reellen Zahl für day_start einen anderen Startzeitpunkt. Die Zahl repräsentiert Bruchteile von Tagen, zum Beispiel 0,125 für 3 Uhr morgens.

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung SET DateFormat in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Funktionsbeispiele

Beispiel	Ergebnis
inday ('01/12/2006 12:23:00 PM', '01/12/2006 12:00:00 AM', 0)	Gibt „wahr“ zurück
inday ('01/12/2006 12:23:00 PM', '01/13/2006 12:00:00 AM', 0)	Gibt „falsch“ zurück
inday ('01/12/2006 12:23:00 PM', '01/12/2006 12:00:00 AM', -1)	Gibt „falsch“ zurück
inday ('01/11/2006 12:23:00 PM', '01/12/2006 12:00:00 AM', -1)	Gibt „wahr“ zurück
inday ('01/12/2006 12:23:00 PM', '01/12/2006 12:00:00 AM', 0, 0.5)	Gibt „falsch“ zurück
inday ('01/12/2006 11:23:00 AM', '01/12/2006 12:00:00 AM', 0, 0.5)	Gibt „wahr“ zurück

Beispiel 1 – load-Anweisung (Skript)

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz, der Transaktionen nach Zeitstempel enthält, wird in eine Tabelle namens `Transactions` geladen.
- Ein Datumsfeld wird im Format `(M/D/YYYY h:mm:ss[.fff] TT)` der Systemvariablen `TimeStamp` bereitgestellt.
- Ein vorangehender load-Befehl enthält die Funktion `inday()`, festgelegt als das Feld `in_day`.

Ladeskript

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Transactions:
    Load
        *,
        inday(date,'01/05/2022 12:00:00 AM', 0) as in_day
        ;
Load
*
Inline
[
id,date,amount
9497,'01/01/2022 7:34:46 PM',13.24
9498,'01/01/2022 10:10:22 PM',31.43
9499,'01/02/2022 8:35:54 AM',36.34
9500,'01/03/2022 2:21:53 PM',51.75
9501,'01/04/2022 6:49:38 PM',15.35
9502,'01/04/2022 10:58:34 PM',74.34
9503,'01/05/2022 5:40:49 AM',73.53
9504,'01/05/2022 11:29:38 AM',50.00
9505,'01/05/2022 7:04:57 PM',47.25
9506,'01/06/2022 8:49:09 AM',74.23
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `in_day`

Ergebnistabelle

date	in_day
01/01/2022 7:34:46 PM	0
01/01/2022 10:10:22 PM	0
01/02/2022 8:35:54 AM	0
01/03/2022 2:21:53 PM	0
01/04/2022 6:49:38 PM	0
01/04/2022 10:58:34 PM	0
01/05/2022 5:40:49 AM	-1
01/05/2022 11:29:38 AM	-1
01/05/2022 7:04:57 PM	-1
01/06/2022 8:49:09 AM	0

Das Feld wird in der vorangehenden load-Anweisung erstellt, indem die Funktion verwendet und das Datumsfeld, ein hartcodierter Zeitstempel für den 5. Januar und ein von 0 als Argumente der Funktion übergeben werden.in_dayinday()period_no0

Beispiel 2 – period_no

Ladeskript und Ergebnisse

Übersicht

Das Ladeskript verwendet den gleichen Datensatz und das gleiche Szenario, die im ersten Beispiel verwendet wurden.

In diesem Beispiel besteht die Aufgabe aber darin, zu berechnen, ob das Transaktionsdatum zwei Tage vor dem 5. Januar lag.

Ladeskript

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Transactions:
    Load
        *,
        inday(date,'01/05/2022 12:00:00 AM', -2) as in_day
        ;
Load
*
Inline
[
id,date,amount
9497,'01/01/2022 7:34:46 PM',13.24
9498,'01/01/2022 10:10:22 PM',31.43
9499,'01/02/2022 8:35:54 AM',36.34
```

```
9500,'01/03/2022 2:21:53 PM',51.75
9501,'01/04/2022 6:49:38 PM',15.35
9502,'01/04/2022 10:58:34 PM',74.34
9503,'01/05/2022 5:40:49 AM',73.53
9504,'01/05/2022 11:29:38 AM',50.00
9505,'01/05/2022 7:04:57 PM',47.25
9506,'01/06/2022 8:49:09 AM',74.23
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `in_day`

Ergebnistabelle

date	in_day
01/01/2022 7:34:46 PM	0
01/01/2022 10:10:22 PM	0
01/02/2022 8:35:54 AM	0
01/03/2022 2:21:53 PM	-1
01/04/2022 6:49:38 PM	0
01/04/2022 10:58:34 PM	0
01/05/2022 5:40:49 AM	0
01/05/2022 11:29:38 AM	0
01/05/2022 7:04:57 PM	0
01/06/2022 8:49:09 AM	0

Da in diesem Beispiel ein `period_no` von -2 als Versatzargument in der Funktion `-2` verwendet wurde, bestimmt die Funktion, ob jedes Transaktionsdatum am 3. Januar stattfand. Dies kann in der Ausgabetabelle überprüft werden, wo eine Transaktion ein boolesches Ergebnis von `inday()` zurückgibt.

Beispiel 3 – day_start

Ladeskript und Ergebnisse

Übersicht

Das Ladeskript verwendet den gleichen Datensatz und das gleiche Szenario, die im vorigen Beispiel verwendet wurden.

In diesem Beispiel beginnt und endet der Arbeitstag laut Unternehmensrichtlinie aber um 7 Uhr.

Ladeskript

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Transactions:
    Load
        *,
        inday(date,'01/05/2022 12:00:00 AM', 0, 7/24) as in_day
        ;
Load
*
Inline
[
id,date,amount
9497,'01/01/2022 7:34:46 PM',13.24
9498,'01/01/2022 10:10:22 PM',31.43
9499,'01/02/2022 8:35:54 AM',36.34
9500,'01/03/2022 2:21:53 PM',51.75
9501,'01/04/2022 6:49:38 PM',15.35
9502,'01/04/2022 10:58:34 PM',74.34
9503,'01/05/2022 5:40:49 AM',73.53
9504,'01/05/2022 11:29:38 AM',50.00
9505,'01/05/2022 7:04:57 PM',47.25
9506,'01/06/2022 8:49:09 AM',74.23
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- date
- in_day

Ergebnistabelle

date	in_day
01/01/2022 7:34:46 PM	0
01/01/2022 10:10:22 PM	0
01/02/2022 8:35:54 AM	0
01/03/2022 2:21:53 PM	0
01/04/2022 6:49:38 PM	-1
01/04/2022 10:58:34 PM	-1
01/05/2022 5:40:49 AM	-1
01/05/2022 11:29:38 AM	0
01/05/2022 7:04:57 PM	0
01/06/2022 8:49:09 AM	0

Da das Argument `start_day` von 7/24, was 7 Uhr morgens ist, in der Funktion `inday()` verwendet wird, bestimmt die Funktion, ob jedes Transaktionsdatum am 4. Januar ab 7 Uhr und am 5. Januar vor 7 Uhr stattfand.

Dies kann in der Ausgabetabelle überprüft werden, wo Transaktionen, die nach 7 Uhr am 4. Januar stattfanden, ein boolesches Ergebnis von zurückgeben, während Transaktionen, die nach 7 Uhr am 5. Januar stattfanden, ein boolesches Ergebnis von zurückgeben.

Beispiel 4 – Diagrammobjekt

Ladeskript und Diagrammformel

Übersicht

Das Ladeskript verwendet den gleichen Datensatz und das gleiche Szenario, die im vorigen Beispiel verwendet wurden.

In diesem Beispiel wird jedoch der unveränderte Datensatz in die Anwendung geladen. Mit der Berechnung bestimmen Sie, ob eine Transaktion am 5. Januar stattfindet, indem Sie eine Kennzahl in einem Diagrammobjekt erstellen.

Ladeskript

```
Transactions:
Load
*
Inline
[
id,date,amount
9497,'01/01/2022 7:34:46 PM',13.24
9498,'01/01/2022 10:10:22 PM',31.43
9499,'01/02/2022 8:35:54 AM',36.34
9500,'01/03/2022 2:21:53 PM',51.75
9501,'01/04/2022 6:49:38 PM',15.35
9502,'01/04/2022 10:58:34 PM',74.34
9503,'01/05/2022 5:40:49 AM',73.53
9504,'01/05/2022 11:29:38 AM',50.00
9505,'01/05/2022 7:04:57 PM',47.25
9506,'01/06/2022 8:49:09 AM',74.23
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu:

- `date`

Um zu berechnen, ob eine Transaktion am 5. Januar stattfindet, erstellen Sie die folgende Kennzahl:

```
=inday(date,'01/05/2022 12:00:00 AM',0)
```

Ergebnistabelle

date	inday(date,'01/05/2022 12:00:00 AM',0)
01/01/2022 7:34:46 PM	0
01/01/2022 10:10:22 PM	0
01/02/2022 8:35:54 AM	0
01/03/2022 2:21:53 PM	0
01/04/2022 6:49:38 PM	0
01/04/2022 10:58:34 PM	0
01/05/2022 5:40:49 AM	-1
01/05/2022 11:29:38 AM	-1
01/05/2022 7:04:57 PM	-1
01/06/2022 8:49:09 AM	0

Beispiel 5 – Szenario

Ladeskript und Ergebnisse

Übersicht

In diesem Beispiel wurde festgestellt, dass aufgrund eines Gerätefehlers die am 5. Januar hergestellten Produkte mangelhaft waren. Der Endbenutzer möchte ein Diagrammobjekt haben, das nach Datum den Status der hergestellten Produkte angibt und zeigt, welche „mangelhaft“ und welche „einwandfrei“ waren und was die am 5. Januar gefertigten Produkte gekostet haben.

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz, der in eine Tabelle namens „Products“ geladen wird.
- Die Tabelle enthält die folgenden Felder:
    - product ID
    - manufacture time
    - cost price

Ladeskript

```
Products:
Load
*
Inline
[
product_id,manufacture_date,cost_price
9497,'01/01/2022 7:34:46 PM',13.24
9498,'01/01/2022 10:10:22 PM',31.43
```

```
9499,'01/02/2022 8:35:54 AM',36.34
9500,'01/03/2022 2:21:53 PM',51.75
9501,'01/04/2022 6:49:38 PM',15.35
9502,'01/04/2022 10:58:34 PM',74.34
9503,'01/05/2022 5:40:49 AM',73.53
9504,'01/05/2022 11:29:38 AM',50.00
9505,'01/05/2022 7:04:57 PM',47.25
9506,'01/06/2022 8:49:09 AM',74.23
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu:

```
=dayname(manufacture_date)
```

Erstellen Sie die folgenden Kennzahlen:

- `=if(only(InDay(manufacture_date,makedate(2022,01,05),0)),'Defective','Faultless')`
- `=sum(cost_price)`

Legen Sie die **Zahlenformatierung** der Kennzahl auf **Währung** fest.

Deaktivieren Sie unter **Darstellung** die Option **Gesamtwerte**.

Ergebnistabelle

dayname (manufacture_date)	=if(only(InDay(manufacture_date,makedate (2022,01,05),0)),'Defective','Faultless')	=sum(cost_ price)
01/01/2022	Faultless	44.67
01/02/2022	Faultless	36.34
01/03/2022	Faultless	51.75
01/04/2022	Faultless	89.69
01/05/2022	Defective	170.78
01/06/2022	Faultless	74.23

Die Funktion `inday()` gibt einen booleschen Wert zurück, wenn sie das Herstellungsdatum der einzelnen Produkte auswertet. Für alle am 5. Januar hergestellten Produkte gibt die Funktion `inday()` einen booleschen Wert von TRUE zurück und markiert die Produkte als „Defective“. Alle Produkte, die einen Wert von FALSE zurückgeben und daher nicht an diesem Tag gefertigt wurden, werden als „Faultless“ markiert.

indaytotime

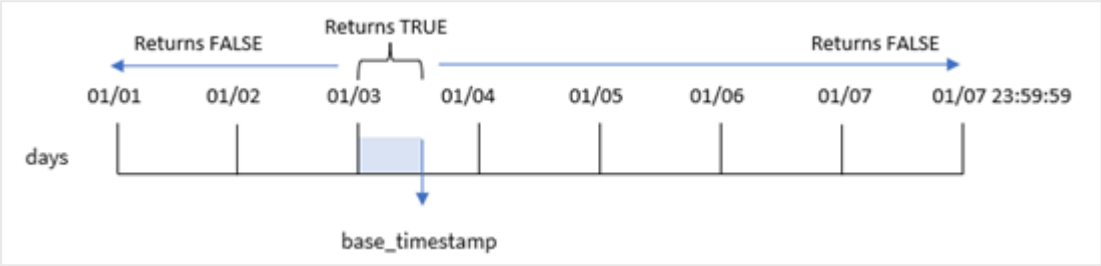
Diese Funktion liefert True, wenn **timestamp** in dem Teil des Tages liegt, der **base_timestamp** enthält, und zwar bis auf die Millisekunde von **base_timestamp**.

Syntax:

```
InDayToTime (timestamp, base_timestamp, period_no[, day_start])
```

Die Funktion `indaytotime()` gibt ein boolesches Ergebnis zurück, abhängig davon, wann ein Zeitstempelwert während des Tagesabschnitts eintritt. Die Startgrenze für dieses Segment ist der Tagesbeginn. Der Standardwert ist Mitternacht, aber dies kann durch das Argument `day_start` der Funktion `indaytotime()` geändert werden. Die Endgrenze des Tagessegments wird von einem Argument `base_timestamp` der Funktion bestimmt.

Diagramm der Funktion `indaytotime`.



Verwendung

Die Funktion `indaytotime()` gibt einen booleschen Wert zurück. In der Regel wird dieser Funktionstyp als Bedingung in einer `if expression` verwendet. Die Funktion `indaytotime()` gibt eine Aggregierung oder Berechnung zurück, abhängig davon, ob ein Zeitstempel in dem Segment des Tages liegt, bis zu und einschließlich der Uhrzeit des Basis-Zeitstempels.

Beispielsweise kann die Funktion `indaytotime()` verwendet werden, um die Summe der Eintrittskartenverkäufe für Vorstellungen zu zeigen, die am heutigen Tag bisher stattgefunden haben.

Rückgabe Datentyp: Boolesch

In Qlik Sense wird der boolesche Wert „wahr“ durch -1 dargestellt, der Wert „falsch“ durch 0.

Argumente

Argument	Beschreibung
`timestamp`	Datum und Uhrzeit, die mit `base_timestamp` verglichen werden sollen.
`base_timestamp`	Datum und Uhrzeit, die zur Interpretation des Zeitstempels verwendet werden.
`period_no`	Mit `period_no` kann ein anderer Tagesbeginn definiert werden. `period_no` ist eine ganze Zahl, wobei 0 für den Tag steht, der `base_timestamp` enthält. Negative Werte von `period_no` stehen für vorangehende, positive Werte für nachfolgende Tage.
`day_start`	(Optional) Wenn Sie mit Tagen arbeiten möchten, die nicht um Mitternacht beginnen, geben Sie einen Versatz als Bruch eines Tages in `day_start` ein. Verwenden Sie beispielsweise 0,125 für 3 Uhr morgens.

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer

Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Funktionsbeispiele

Beispiel	Ergebnis
`indaytotime` ('01/12/2006 12:23:00 PM', '01/12/2006 11:59:00 PM', 0)	Gibt „wahr“ zurück
`indaytotime` ('01/12/2006 12:23:00 PM', '01/12/2006 12:00:00 AM', 0)	Gibt „falsch“ zurück
`indaytotime` '01/11/2006 12:23:00 PM', '01/12/2006 11:59:00 PM', -1)	Gibt „wahr“ zurück

Beispiel 1 – keine zusätzlichen Argumente

Ladeskript und Ergebnisse

Überblick

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz, der eine Reihe von Transaktionen für den Zeitraum zwischen dem 4. und 5. Januar enthält und in eine Tabelle mit dem Namen „Transactions“ geladen wird.
- Ein Datumsfeld, das im Format `(M/D/YYYY h:mm:ss[.fff] TT)` der Systemvariablen `TimeStamp` bereitgestellt wird.
- Ein vorangehender load-Befehl, der die Funktion `indaytotime()` enthält, die auf `'in_day_to_time'` festgelegt ist. Dieses Feld bestimmt für jede Transaktion, ob sie vor 9:00 Uhr stattfindet.

Ladeskript

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Transactions:
    Load
        *,
        indaytotime(date,'01/05/2022 9:00:00 AM',0) as in_day_to_time
        ;
Load
*
Inline
[
id,date,amount
8188,'01/04/2022 3:41:54 AM',25.66
```

```
8189,'01/04/2022 4:19:43 AM',87.21
8190,'01/04/2022 4:53:47 AM',53.80
8191,'01/04/2022 8:38:53 AM',69.98
8192,'01/04/2022 10:37:52 AM',57.42
8193,'01/04/2022 1:54:10 PM',45.89
8194,'01/04/2022 5:53:23 PM',82.77
8195,'01/04/2022 8:13:26 PM',36.23
8196,'01/04/2022 10:00:49 PM',76.11
8197,'01/05/2022 7:45:37 AM',82.06
8198,'01/05/2022 8:44:36 AM',17.17
8199,'01/05/2022 11:26:08 AM',40.39
8200,'01/05/2022 6:43:08 PM',37.23
8201,'01/05/2022 10:54:10 PM',88.27
8202,'01/05/2022 11:09:09 PM',95.93
];
```

Ergebnisse

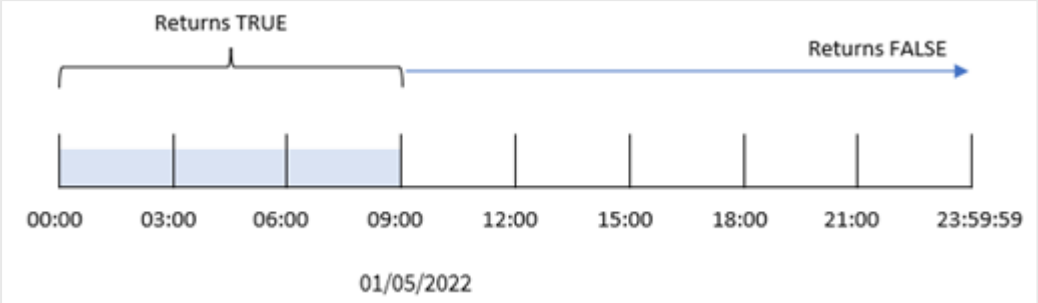
Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `in_day_to_time`

Ergebnistabelle

Datum	in_day_to_time
01/04/2022 3:41:54 AM	0
01/04/2022 4:19:43 AM	0
01/04/2022 04:53:47 AM	0
01/04/2022 8:38:53 AM	0
01/04/2022 10:37:52 AM	0
01/04/2022 1:54:10 PM	0
01/04/2022 5:53:23 PM	0
01/04/2022 8:13:26 PM	0
01/04/2022 10:00:49 PM	0
01/05/2022 7:45:37 AM	-1
01/05/2022 8:44:36 AM	-1
01/05/2022 11:26:08 AM	0
01/05/2022 6:43:08 PM	0
01/05/2022 10:54:10 PM	0
01/05/2022 11:09:09 PM	0

Beispiel 1: Diagramm der Funktion `indaytotime` *mit 9:00-Uhr-Limit.*



Das `in_day_to_time field` wird in der vorangegangenen load-Anweisung erstellt, indem die Funktion `indaytotime()` verwendet und das Datumsfeld, ein hartcodierter Zeitstempel für 9:00 Uhr 5. Januar und ein Versatz von 0 als Argumente der Funktion übergeben werden. Alle Transaktionen, die zwischen Mitternacht und 9:00 Uhr am 5. Januar erfolgen, geben WAHR zurück.

Beispiel 2 – period_no

Ladeskript und Ergebnisse

Übersicht

Das Ladeskript verwendet den gleichen Datensatz und das gleiche Szenario, die im ersten Beispiel verwendet wurden.

In diesem Beispiel berechnen Sie aber, ob das Transaktionsdatum einen Tag vor 9:00 Uhr am 5. Januar lag.

Ladeskript

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Transactions:
    Load
        *,
        indaytotime(date,'01/05/2022 9:00:00 AM', -1) as in_day_to_time
        ;
Load
*
Inline
[
id,date,amount
8188,'01/04/2022 3:41:54 AM',25.66
8189,'01/04/2022 4:19:43 AM',87.21
8190,'01/04/2022 4:53:47 AM',53.80
8191,'01/04/2022 8:38:53 AM',69.98
8192,'01/04/2022 10:37:52 AM',57.42
8193,'01/04/2022 1:54:10 PM',45.89
8194,'01/04/2022 5:53:23 PM',82.77
8195,'01/04/2022 8:13:26 PM',36.23
8196,'01/04/2022 10:00:49 PM',76.11
8197,'01/05/2022 7:45:37 AM',82.06
8198,'01/05/2022 8:44:36 AM',17.17
```

```
8199,'01/05/2022 11:26:08 AM',40.39
8200,'01/05/2022 6:43:08 PM',37.23
8201,'01/05/2022 10:54:10 PM',88.27
8202,'01/05/2022 11:09:09 PM',95.93
];
```

Ergebnisse

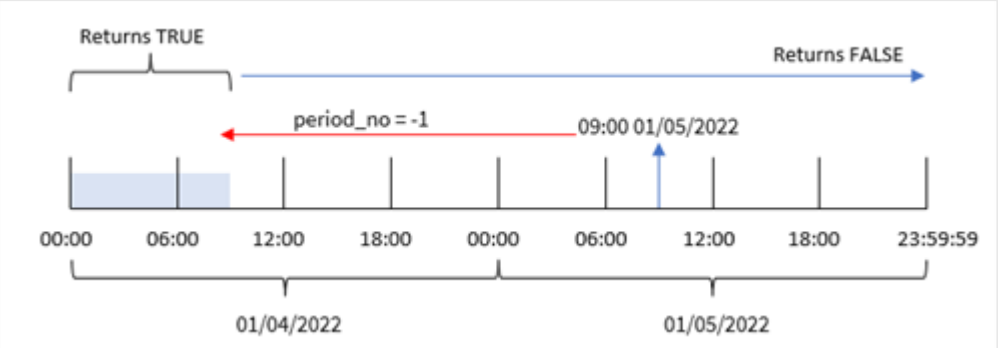
Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `in_day_to_time`

Ergebnistabelle

Datum	in_day_to_time
01/04/2022 3:41:54 AM	-1
01/04/2022 4:19:43 AM	-1
01/04/2022 04:53:47 AM	-1
01/04/2022 8:38:53 AM	-1
01/04/2022 10:37:52 AM	0
01/04/2022 1:54:10 PM	0
01/04/2022 5:53:23 PM	0
01/04/2022 8:13:26 PM	0
01/04/2022 10:00:49 PM	0
01/05/2022 7:45:37 AM	0
01/05/2022 8:44:36 AM	0
01/05/2022 11:26:08 AM	0
01/05/2022 6:43:08 PM	0
01/05/2022 10:54:10 PM	0
01/05/2022 11:09:09 PM	0

Beispiel 2: Diagramm der Funktion `indaytotime` mit Transaktionen ab dem 4. Januar



Da in diesem Beispiel ein Versatz von -1 als Versatzargument in der Funktion `indaytotime()` verwendet wurde, bestimmt die Funktion, ob jedes Transaktionsdatum vor 9:00 Uhr am 4. Januar stattfand. Dies kann in der Ausgabetabelle überprüft werden, wo eine Transaktion ein boolesches Ergebnis von WAHR zurückgibt.

Beispiel 3 – day_start

Ladeskript und Ergebnisse

Übersicht

Es werden derselbe Datenatz und dasselbe Szenario wie im ersten Beispiel verwendet.

In diesem Beispiel beginnt und endet der Arbeitstag laut Unternehmensrichtlinie aber um 8 Uhr.

Ladeskript

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Transactions:
    Load
        *,
        indaytotime(date,'01/05/2022 9:00:00 AM', 0,8/24) as in_day_to_time
        ;
Load
*
Inline
[
id,date,amount
8188,'01/04/2022 3:41:54 AM',25.66
8189,'01/04/2022 4:19:43 AM',87.21
8190,'01/04/2022 4:53:47 AM',53.80
8191,'01/04/2022 8:38:53 AM',69.98
8192,'01/04/2022 10:37:52 AM',57.42
8193,'01/04/2022 1:54:10 PM',45.89
8194,'01/04/2022 5:53:23 PM',82.77
8195,'01/04/2022 8:13:26 PM',36.23
8196,'01/04/2022 10:00:49 PM',76.11
8197,'01/05/2022 7:45:37 AM',82.06
8198,'01/05/2022 8:44:36 AM',17.17
8199,'01/05/2022 11:26:08 AM',40.39
```

```
8200,'01/05/2022 6:43:08 PM',37.23
8201,'01/05/2022 10:54:10 PM',88.27
8202,'01/05/2022 11:09:09 PM',95.93
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `in_day_to_time`

Ergebnistabelle

Datum	in_day_to_time
01/04/2022 3:41:54 AM	0
01/04/2022 4:19:43 AM	0
01/04/2022 04:53:47 AM	0
01/04/2022 8:38:53 AM	0
01/04/2022 10:37:52 AM	0
01/04/2022 1:54:10 PM	0
01/04/2022 5:53:23 PM	0
01/04/2022 8:13:26 PM	0
01/04/2022 10:00:49 PM	0
01/05/2022 7:45:37 AM	0
01/05/2022 8:44:36 AM	-1
01/05/2022 11:26:08 AM	0
01/05/2022 6:43:08 PM	0
01/05/2022 10:54:10 PM	0
01/05/2022 11:09:09 PM	0

Beispiel 3: Diagramm der Funktion `indaytotime` mit Transaktionen von 8:00 bis 9:00 Uhr.

Da das Argument `start_day` von 8/24, das 8:00 Uhr morgens entspricht, in der Funktion `indaytotime()` verwendet wird, beginnt und endet jeder Tag um 8:00 Uhr. Daher gibt die Funktion `indaytotime()` ein boolesches Ergebnis von TRUE für jede Transaktion zurück, die am 5. Januar zwischen 8:00 und 9:00 Uhr stattgefunden hat.

Beispiel 4 – Diagrammobjekt

Ladeskript und Diagrammformel

Überblick

Es werden derselbe Datensatz und dasselbe Szenario wie im ersten Beispiel verwendet.

In diesem Beispiel wird jedoch der unveränderte Datensatz in die Anwendung geladen. Mit der Berechnung bestimmen Sie, ob eine Transaktion am 5. Januar vor 9:00 Uhr stattfindet, indem Sie eine Kennzahl in einem Diagrammobjekt erstellen.

Ladeskript

```
Transactions:
Load
*
Inline
[
id,date,amount
8188,'01/04/2022 3:41:54 AM',25.66
8189,'01/04/2022 4:19:43 AM',87.21
8190,'01/04/2022 4:53:47 AM',53.80
8191,'01/04/2022 8:38:53 AM',69.98
8192,'01/04/2022 10:37:52 AM',57.42
8193,'01/04/2022 1:54:10 PM',45.89
8194,'01/04/2022 5:53:23 PM',82.77
8195,'01/04/2022 8:13:26 PM',36.23
8196,'01/04/2022 10:00:49 PM',76.11
8197,'01/05/2022 7:45:37 AM',82.06
8198,'01/05/2022 8:44:36 AM',17.17
8199,'01/05/2022 11:26:08 AM',40.39
8200,'01/05/2022 6:43:08 PM',37.23
8201,'01/05/2022 10:54:10 PM',88.27
8202,'01/05/2022 11:09:09 PM',95.93
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu:

`date`durch.

Um zu bestimmen , ob eine Transaktion am 5. Januar vor 9:00 Uhr stattfindet, erstellen Sie die folgende Kennzahl:

```
=indaytotime(date,'01/05/2022 9:00:00 AM',0)
```

Ergebnistabelle

Datum	**=indaytotime(date,'01/05/2022 9:00:00 AM',0)**
01/04/2022 3:41:54 AM	0
01/04/2022 4:19:43 AM	0
01/04/2022 04:53:47 AM	0
01/04/2022 8:38:53 AM	0
01/04/2022 10:37:52 AM	0
01/04/2022 1:54:10 PM	0
01/04/2022 5:53:23 PM	0
01/04/2022 8:13:26 PM	0
01/04/2022 10:00:49 PM	0
01/05/2022 7:45:37 AM	-1
01/05/2022 8:44:36 AM	-1
01/05/2022 11:26:08 AM	0
01/05/2022 6:43:08 PM	0
01/05/2022 10:54:10 PM	0
01/05/2022 11:09:09 PM	0

Die Kennzahl `in_day_to_time` wird im Diagrammobjekt erstellt, indem die Funktion `indaytotime()` verwendet und das Datumsfeld, ein hartcodierter Zeitstempel für 5. Januar 9:00 Uhr und ein Versatz von `0` als Argumente der Funktion übergeben werden. Alle Transaktionen, die zwischen Mitternacht und 9:00 Uhr am 5. Januar erfolgen, geben WAHR zurück. Dies wird in der Ergebnistabelle validiert.

Beispiel 5 – Szenario

Ladeskript und Ergebnisse

Übersicht

In diesem Beispiel wird ein Datensatz für Eintrittskartenverkäufe für ein örtliches Kino in eine Tabelle namens Ticket_Sales geladen. Heute ist der 3. Mai 2022, und es ist 11:00 Uhr.

In diesem Fall wünscht sich der Benutzer ein KPI-Diagrammobjekt, das den Umsatz aus allen Vorstellungen zeigt, die am heutigen Tag bisher stattgefunden haben.

Ladeskript

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Ticket_Sales:
Load
*
Inline
```

```
[
sale ID, show time, ticket price
1,05/01/2022 09:30:00 AM,10.50
2,05/03/2022 05:30:00 PM,21.00
3,05/03/2022 09:30:00 AM,10.50
4,05/03/2022 09:30:00 AM,31.50
5,05/03/2022 09:30:00 AM,10.50
6,05/03/2022 12:00:00 PM,42.00
7,05/03/2022 12:00:00 PM,10.50
8,05/03/2022 05:30:00 PM,42.00
9,05/03/2022 08:00:00 PM,31.50
10,05/04/2022 10:30:00 AM,31.50
11,05/04/2022 12:00:00 PM,10.50
12,05/04/2022 05:30:00 PM,10.50
13,05/05/2022 05:30:00 PM,21.00
14,05/06/2022 12:00:00 PM,21.00
15,05/07/2022 09:30:00 AM,42.00
16,05/07/2022 10:30:00 AM,42.00
17,05/07/2022 10:30:00 AM,10.50
18,05/07/2022 05:30:00 PM,10.50
19,05/08/2022 05:30:00 PM,21.00
20,05/11/2022 09:30:00 AM,10.50
];
```

Ergebnisse

Gehen Sie folgendermaßen vor:

1. Erstellen Sie ein KPI-Objekt.
2. Erstellen Sie eine Kennzahl, die die Summe aller Eintrittskartenverkäufe für Vorstellungen zeigt, die am heutigen Tag bisher stattgefunden haben. Verwenden Sie hierfür die Funktion `indaytotime()`:

```
=sum(if(indaytotime([show time],'05/03/2022 11:00:00 AM',0),[ticket price],0))
```

3. Erstellen Sie eine Beschriftung für das KPI-Objekt, „Current Revenue“.
4. Legen Sie die **Zahlenformatierung** der Kennzahl auf **Währung** fest.

Die Summe der Eintrittskartenverkäufe bis 11:00 Uhr am 3. Mai 2022 beträgt 52,50 $.

Die Funktion `indaytotime ()` gibt einen booleschen Wert zurück, wenn die Vorstellungszeiten für jeden der Eintrittskartenverkäufe mit der aktuellen Uhrzeit verglichen werden ('05/03/2022 11:00:00 AM'). Für jede Vorstellung am 3. Mai vor 11:00 Uhr gibt die Funktion `indaytotime ()` einen booleschen Wert von TRUE zurück, und der Eintrittskartenpreis wird in die Gesamtsumme eingeschlossen.

inlunarweek

Diese Funktion bestimmt, ob **timestamp** innerhalb der Mondwoche liegt, die **base_date** enthält. Bei Mondwochen in Qlik Sense wird der 1. Januar als der erste Tag der Woche gezählt. Mit Ausnahme der letzten Woche des Jahres umfasst jede Woche genau sieben Tage.

Syntax:

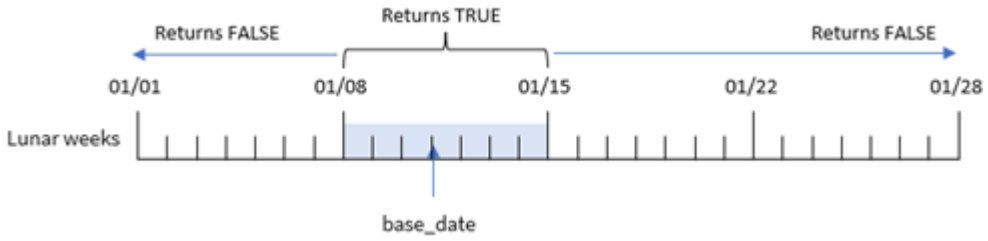
```
InLunarWeek (timestamp, base_date, period_no[, first_week_day])
```

Rückgabe Datentyp: Boolesch

In Qlik Sense wird der boolesche Wert „wahr“ durch -1 dargestellt, der Wert „falsch“ durch 0.

Die Funktion `inlunarweek()` bestimmt, in welche Mondwoche das `base_date` fällt. Dann gibt sie ein boolesches Ergebnis zurück, sobald sie ermittelt hat, ob jeder Zeitstempelwert in die gleiche Mondwoche wie das `base_date` fällt.

Diagramm der Funktion `inlunarweek()`



Verwendung

Die Funktion `inlunarweek()` gibt einen booleschen Wert zurück. In der Regel wird dieser Funktionstyp als Bedingung in einem IF-Ausdruck verwendet. Damit wird eine Aggregierung oder eine Berechnung zurückgegeben, abhängig davon, ob das ausgewertete Datum in die betreffende Mondwoche fällt.

Beispielsweise kann die Funktion `inlunarweek()` verwendet werden, um alle in einer bestimmten Mondwoche gefertigten Geräte zu identifizieren.

Argumente

Argument	Beschreibung
timestamp	Das Datum, das mit **base_date** verglichen werden soll.
base_date	Datum, das für die Interpretation der Mondwoche verwendet wird.
period_no	Mit **period_no** kann ein anderer Beginn für die Mondwoche definiert werden. period_no ist eine ganze Zahl, wobei 0 für die Mondwoche steht, die **base_date** enthält. Negative Werte von **period_no** stehen für vorangehende, positive Werte für nachfolgende Mondwochen.
first_week_day	Ein Startwert, der größer oder kleiner als Null sein kann. Dadurch wird der Beginn des Jahres um die angegebene Anzahl an Tagen und/oder den Bruchteil eines Tages verschoben.

Funktionsbeispiele

Beispiel	Ergebnis
`inlunarweek ('01/12/2013', '01/14/2013', 0)`	Gibt `TRUE` zurück, da der Wert von `timestamp`, 01/12/2013, in die Woche vom 01/08/2013 bis zum 01/14/2013 fällt.

Beispiel	Ergebnis
`inlunarweek ('01/12/2013', '01/07/2013', 0)`	Gibt FALSE zurück, da das `base_date`, 01/07/2013, in der Mondwoche liegt, die als 01/01/2013 bis 01/07/2013 definiert ist.
`inlunarweek ('01/12/2013', '01/14/2013', -1)`	Gibt FALSE zurück. Durch Angabe eines Werts von period_no als -1 wird die Woche auf die vorhergehende Woche verschoben, vom 01/01/2013 bis zum 01/07/2013.
`inlunarweek ('01/07/2013', 01/14/2013', -1)`	Gibt TRUE zurück. Verglichen mit dem vorherigen Beispiel befindet sich `timestamp` in der darauffolgenden Woche, nachdem die Verschiebung nach hinten berücksichtigt wurde.
`inlunarweek ('01/11/2006', '01/08/2006', 0, 3)`	Gibt FALSE zurück. Die Angabe eines Werts von 3 für `first_week_day` bedeutet, dass der Jahresbeginn ab dem 01/04/2013 berechnet wird. Daher fällt der Wert von `base_date` in die erste Woche, und der Wert von `timestamp` fällt in die Woche vom 01/11/2013 bis zum 01/17/2013.

Die Funktion `inlunarweek()` wird oft in Kombination mit den folgenden Funktionen verwendet:

Verwandte Funktionen

Funktion	Interaktion
lunarweekname (page 884)	Diese Funktion wird verwendet, um die Mondwochennummer des Jahres zu bestimmen, in dem ein Eingabedatum liegt.

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Beispiel 1 – keine zusätzlichen Argumente

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz mit Transaktionen für den Monat Januar wird in eine Tabelle namens `Transactions` geladen.
- Das Datumsfeld wurde im Format der Systemvariablen `DateFormat` (MM/TT/JJJJ) bereitgestellt.

Erstellen Sie ein Feld `in_lunar_week`, das bestimmt, ob die Transaktionen in der gleichen Mondwoche wie der 10. Januar stattfanden.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        inlunarweek(date,'01/10/2022', 0) as in_lunar_week
        ;
Load
*
Inline
[
id,date,amount
8183,'1/5/2022',42.32
8184,'1/6/2022',68.22
8185,'1/7/2022',15.25
8186,'1/8/2022',25.26
8187,'1/9/2022',37.23
8188,'1/10/2022',37.23
8189,'1/11/2022',17.17
8190,'1/12/2022',88.27
8191,'1/13/2022',57.42
8192,'1/14/2022',53.80
8193,'1/15/2022',82.06
8194,'1/16/2022',87.21
8195,'1/17/2022',95.93
8196,'1/18/2022',45.89
8197,'1/19/2022',36.23
8198,'1/20/2022',25.66
8199,'1/21/2022',82.77
8200,'1/22/2022',69.98
8201,'1/23/2022',76.11
];
```

Ergebnisse

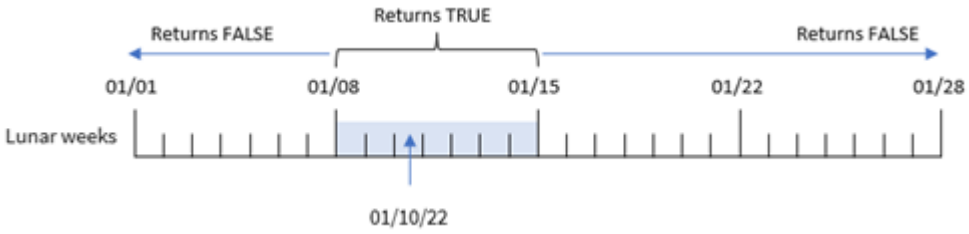
Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `in_lunar_week`

Ergebnistabelle

date	in_lunar_week
1/5/2022	0
1/6/2022	0
1/7/2022	0
1/8/2022	-1
1/9/2022	-1
1/10/2022	-1
1/11/2022	-1
1/12/2022	-1
1/13/2022	-1
1/14/2022	-1
1/15/2022	0
1/16/2022	0
1/17/2022	0
1/18/2022	0
1/19/2022	0
1/20/2022	0
1/21/2022	0
1/22/2022	0
1/23/2022	0

Funktion `inlunarweek()`, einfaches Beispiel



Das Feld `in_lunar_week` wird im vorangehenden load-Befehl erstellt, indem die Funktion `inlunarweek()` verwendet und Folgendes als Argumente der Funktion übergeben wird:

- Das Feld `date`
- Ein hartcodiertes Datum, der 10. Januar, als `base_date`
- Eine `period_no` von 0.

Da Mondwochen am 1. Januar beginnen, fällt der 10. Januar in die Mondwoche, die am 8. Januar beginnt und am 14. Januar endet. Daher gibt jede Transaktion, die zwischen diesen beiden Datumswerten im Januar stattfindet, einen booleschen Wert von TRUE zurück. Dies wird in der Ergebnistabelle validiert.

Beispiel 2 – period_no

Beispiele und Ergebnisse:

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Derselbe Datensatz und dasselbe Szenario wie im ersten Beispiel.
- Das Datumsfeld wurde im Format der Systemvariablen `DateFormat` (MM/TT/JJJJ) bereitgestellt.

In diesem Beispiel besteht die Aufgabe aber darin, ein Feld `2_lunar_weeks_later` zu erstellen, das bestimmt, ob die Transaktionen zwei Mondwochen nach dem 10. Januar stattfanden oder nicht.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        inlunarweek(date,'01/10/2022', 2) as [2_lunar_weeks_later]
        ;
Load
*
Inline
[
id,date,amount
8183,'1/5/2022',42.32
8184,'1/6/2022',68.22
8185,'1/7/2022',15.25
8186,'1/8/2022',25.26
8187,'1/9/2022',37.23
8188,'1/10/2022',37.23
8189,'1/11/2022',17.17
8190,'1/12/2022',88.27
8191,'1/13/2022',57.42
8192,'1/14/2022',53.80
8193,'1/15/2022',82.06
8194,'1/16/2022',87.21
8195,'1/17/2022',95.93
8196,'1/18/2022',45.89
8197,'1/19/2022',36.23
8198,'1/20/2022',25.66
8199,'1/21/2022',82.77
8200,'1/22/2022',69.98
```

```
8201,'1/23/2022',76.11
];
```

Ergebnisse

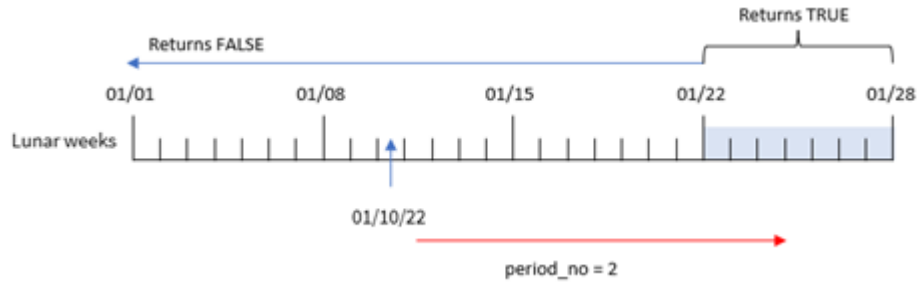
Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `2_lunar_weeks_later`

Ergebnistabelle

date	**2_lunar_weeks_later**
1/5/2022	0
1/6/2022	0
1/7/2022	0
1/8/2022	0
1/9/2022	0
1/10/2022	0
1/11/2022	0
1/12/2022	0
1/13/2022	0
1/14/2022	0
1/15/2022	0
1/16/2022	0
1/17/2022	0
1/18/2022	0
1/19/2022	0
1/20/2022	0
1/21/2022	0
1/22/2022	-1
1/23/2022	-1

Funktion `inlunarweek()`, Beispiel „period_no“



Da in diesem Beispiel eine `period_no` von 2 als Versatzargument in der Funktion `inlunarweek()` verwendet wurde, definiert die Funktion die Woche, die am 22. Januar beginnt, als die Mondwoche, für die Transaktionen validiert werden. Daher gibt jede Transaktion, die zwischen dem 22. Januar und dem 28. Januar stattfindet, ein boolesches Ergebnis von `TRUE` zurück.

Beispiel 3 – first_week_day

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript verwendet den gleichen Datensatz und das gleiche Szenario wie das erste Beispiel. In diesem Beispiel wird aber für den Beginn der Mondwochen der 6. Januar festgelegt.

- Derselbe Datensatz und dasselbe Szenario wie im ersten Beispiel.
- Die `DateFormat`-Standardsystemvariable MM/TT/JJJJ wird verwendet.
- Ein Argument für `first_week_day` von 5. Damit wird der Mondwochenbeginn auf den 5. Januar festgelegt.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        inlunarweek(date,'01/10/2022', 0,5) as in_lunar_week
        ;
Load
*
Inline
[
id,date,amount
8183,'1/5/2022',42.32
8184,'1/6/2022',68.22
8185,'1/7/2022',15.25
8186,'1/8/2022',25.26
```

```
8187,'1/9/2022',37.23
8188,'1/10/2022',37.23
8189,'1/11/2022',17.17
8190,'1/12/2022',88.27
8191,'1/13/2022',57.42
8192,'1/14/2022',53.80
8193,'1/15/2022',82.06
8194,'1/16/2022',87.21
8195,'1/17/2022',95.93
8196,'1/18/2022',45.89
8197,'1/19/2022',36.23
8198,'1/20/2022',25.66
8199,'1/21/2022',82.77
8200,'1/22/2022',69.98
8201,'1/23/2022',76.11
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

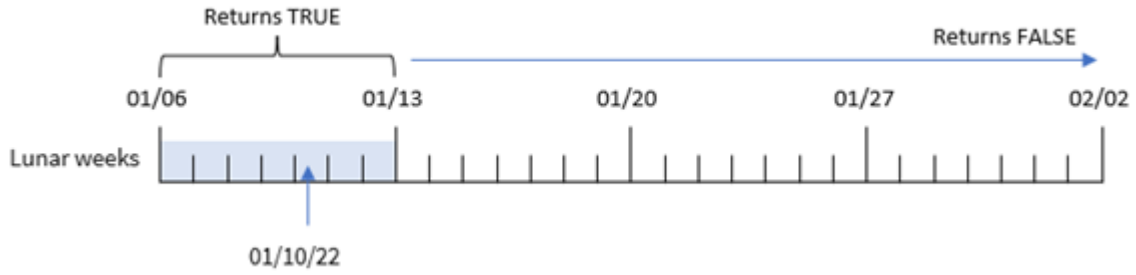
- `date`
- `in_lunar_week`

Ergebnistabelle

date	**in_lunar_week**
1/5/2022	0
1/6/2022	-1
1/7/2022	-1
1/8/2022	-1
1/9/2022	-1
1/10/2022	-1
1/11/2022	-1
1/12/2022	-1
1/13/2022	0
1/14/2022	0
1/15/2022	0
1/16/2022	0
1/17/2022	0
1/18/2022	0

date	in_lunar_week
1/19/2022	0
1/20/2022	0
1/21/2022	0
1/22/2022	0
1/23/2022	0

Funktion `inlunarweek()`, Beispiel „first_week_day“



Da in diesem Fall das Argument `first_week_date` von 5 in der Funktion `inlunarweek()` verwendet wird, wird der Beginn des Mondwochenkalenders auf den 6. Januar verschoben. Somit fällt der 10. Januar in die Mondwoche, die am 6. Januar beginnt und am 12. Januar endet. Jede Transaktion, die zwischen diesen beiden Datumswerten liegt, gibt einen booleschen Wert von `TRUE` zurück.

Beispiel 4 – Diagrammobjekt

Ladeskript und Diagrammformel:

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Derselbe Datensatz und dasselbe Szenario wie im ersten Beispiel.
- Das Datumsfeld wurde im Format der Systemvariablen `DateFormat` (MM/TT/JJJJ) bereitgestellt.

In diesem Beispiel wird jedoch der unveränderte Datensatz in die Anwendung geladen. Die Berechnung, die bestimmt, ob Transaktionen in der gleichen Mondwoche wie der 10. Januar stattfanden, wird als Kennzahl in einem Diagrammobjekt der Anwendung erstellt.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
```

```
*
Inline
[
id,date,amount
8183,'1/5/2022',42.32
8184,'1/6/2022',68.22
8185,'1/7/2022',15.25
8186,'1/8/2022',25.26
8187,'1/9/2022',37.23
8188,'1/10/2022',37.23
8189,'1/11/2022',17.17
8190,'1/12/2022',88.27
8191,'1/13/2022',57.42
8192,'1/14/2022',53.80
8193,'1/15/2022',82.06
8194,'1/16/2022',87.21
8195,'1/17/2022',95.93
8196,'1/18/2022',45.89
8197,'1/19/2022',36.23
8198,'1/20/2022',25.66
8199,'1/21/2022',82.77
8200,'1/22/2022',69.98
8201,'1/23/2022',76.11
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu: `date`.

Um zu berechnen, ob eine Transaktion in der Mondwoche stattfindet, die den 10. Januar enthält, erstellen Sie die folgende Kennzahl:

```
= inlunarweek(date,'01/10/2022', 0)
```

Ergebnistabelle

date	**=inlunarweek(date,'01/10/2022', 0)**
1/5/2022	0
1/6/2022	0
1/7/2022	0
1/8/2022	-1
1/9/2022	-1
1/10/2022	-1
1/11/2022	-1
1/12/2022	-1
1/13/2022	-1

date	=inlunarweek(date,'01/10/2022', 0)
1/14/2022	-1
1/15/2022	0
1/16/2022	0
1/17/2022	0
1/18/2022	0
1/19/2022	0
1/20/2022	0
1/21/2022	0
1/22/2022	0
1/23/2022	0

Beispiel 5 – Szenario

Ladeskript und Diagrammformel:

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Datensatz, der in eine Tabelle namens `Products` geladen wird
- Informationen bestehend aus Produkt-ID, Herstellungsdatum und Selbstkosten

Es wurde festgestellt, dass aufgrund eines Ausrüstungsfehler die Produkte, die in der Mondwoche des 12. Januar hergestellt wurden, mangelhaft waren. Der Endbenutzer möchte ein Diagrammobjekt, das nach Namen der Mondwoche den Status der hergestellten Produkte angibt und zeigt, welche „mangelhaft“ und welche „einwandfrei“ waren und was die in diesem Monat gefertigten Produkte gekostet haben.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
product_id,manufacture_date,cost_price
8183,'1/5/2022',42.32
8184,'1/6/2022',68.22
8185,'1/7/2022',15.25
8186,'1/8/2022',25.26
8187,'1/9/2022',37.23
```

```
8188,'1/10/2022',37.23
8189,'1/11/2022',17.17
8190,'1/12/2022',88.27
8191,'1/13/2022',57.42
8192,'1/14/2022',53.80
8193,'1/15/2022',82.06
8194,'1/16/2022',87.21
8195,'1/17/2022',95.93
8196,'1/18/2022',45.89
8197,'1/19/2022',36.23
8198,'1/20/2022',25.66
8199,'1/21/2022',82.77
8200,'1/22/2022',69.98
8201,'1/23/2022',76.11
];
```

Ergebnisse

Gehen Sie folgendermaßen vor:

1. Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle.
2. Erstellen Sie eine Dimension, um die Monatsnamen anzuzeigen:
   `=lunarweekname(manufacture_date)`
3. Erstellen Sie eine Kennzahl, um zu identifizieren, welche Produkte mangelhaft und welche einwandfrei sind. Verwenden Sie dafür die Funktion `inlunarweek()`:
   `=if(only(inlunarweek(manufacture_date,makedate(2022,01,12),0)), 'Defective','Faultless')`
4. Erstellen Sie eine Kennzahl zum Summieren des `cost_price` der Produkte:
   `=sum(cost_price)`
5. Legen Sie das **Zahlenformat** der Kennzahl auf **Währung** fest.
6. Deaktivieren Sie unter **Darstellung** die Option **Gesamtwerte**.

Ergebnistabelle

lunarweekname (manufacture_date)	=if(only(inlunarweek(manufacture_date,makedate (2022,01,12),0)), 'Defective','Faultless')	sum(cost_ price)
2022/01	Faultless	$125.79
2022/02	Defective	$316.38
2022/03	Faultless	$455.75
2022/04	Faultless	$146.09

Die Funktion `inlunarweek()` gibt einen booleschen Wert zurück, wenn sie das Herstellungsdatum der einzelnen Produkte auswertet. Für alle in der Mondwoche des 10. Januar hergestellten Produkte gibt die Funktion `inlunarweek()` einen booleschen Wert von TRUE zurück und markiert die Produkte als „mangelhaft“. Alle Produkte, die einen Wert von FALSE zurückgeben und daher nicht in der betreffenden Woche gefertigt wurden, werden als „einwandfrei“ markiert.

inlunarweektodate

Diese Funktion gibt an, ob **timestamp** in dem Teil der Mondwoche liegt, die **base_date** enthält, und zwar bis einschließlich der letzten Millisekunde davon. Bei Mondwochen in Qlik Sense wird der 1. Januar als der erste Tag der Woche gezählt. Mit Ausnahme der letzten Woche des Jahres umfasst jede Woche genau sieben Tage.

Syntax:

```
InLunarWeekToDate (timestamp, base_date, period_no [, first_week_day])
```

Rückgabe Datentyp: Boolesch

> *In Qlik Sense wird der boolesche Wert „wahr“ durch -1 dargestellt, der Wert „falsch“ durch 0.*

Beispieldiagramm der Funktion `inlunarweektodate()`

Die Funktion `inlunarweektodate()` verhält sich als Endpunkt der Mondwoche. Im Gegensatz dazu bestimmt die Funktion `inlunarweek()`, in welche Mondwoche das `base_date` fällt. Wenn beispielsweise das `base_date` der 5. Januar ist, gibt jeder Zeitstempel zwischen dem 1. und 5. Januar einen booleschen Wert von TRUE zurück, während Datumswerte am 6. und 7. Januar oder später ein boolesches Ergebnis von FALSE zurückgeben.

Argumente

Argument	Beschreibung
timestamp	Das Datum, das mit **base_date** verglichen werden soll.
base_date	Datum, das für die Interpretation der Mondwoche verwendet wird.
period_no	Mit **period_no** kann ein anderer Beginn für die Mondwoche definiert werden. period_no ist eine ganze Zahl, wobei 0 für die Mondwoche steht, die **base_date** enthält. Negative Werte von **period_no** stehen für vorangehende, positive Werte für nachfolgende Mondwochen.
first_week_day	Ein Startwert, der größer oder kleiner als Null sein kann. Dadurch wird der Beginn des Jahres um die angegebene Anzahl an Tagen und/oder den Bruchteil eines Tages verschoben.

Verwendung

Die Funktion `inlunarweektodate()` gibt einen booleschen Wert zurück. In der Regel wird dieser Funktionstyp als Bedingung in einem IF-Ausdruck verwendet. Die Funktion `inlunarweektodate()` kann dann verwendet werden, wenn der Benutzer möchte, dass die Berechnung eine Aggregierung oder eine Berechnung zurückgibt, je nachdem, ob das ausgewertete Datum in einem bestimmten Segment der betreffenden Woche lag.

Beispielsweise kann die Funktion `inlunarweektodate()` verwendet werden, um alle in einer bestimmten Woche bis zu einem bestimmten Datum gefertigten Geräte zu identifizieren.

Funktionsbeispiele

Beispiel	**Ergebnis**
`inlunarweektodate ('01/12/2013', '01/13/2013', 0)`	Gibt `TRUE` zurück, da der Wert von `timestamp`, 01/12/2013, in den Teil der Woche vom 01/08/2013 bis zum 01/13/2013 fällt.
`inlunarweektodate ('01/12/2013', '01/11/2013', 0)`	Gibt `FALSE` zurück, weil der Wert von `timestamp` nach dem Wert `base_date` liegt, obwohl die beiden Datumswerte in derselben Mondwoche vor dem 01/12/2012 liegen.
`inlunarweektodate ('01/12/2006', '01/05/2006', 1)`	Gibt `TRUE` zurück. Durch die Angabe eines Werts von 1 für `period_no` wird `base_date` eine Woche nach vorne verschoben, sodass der Wert von `timestamp` in den Teil der Mondwoche fällt.

Die Funktion `inlunarweektodate()` wird oft in Kombination mit den folgenden Funktionen verwendet:

Verwandte Funktionen

Funktion	**Interaktion**
lunarweekname (page 884)	Diese Funktion wird verwendet, um die Mondwochennummer des Jahres zu bestimmen, in dem ein Eingabedatum liegt.

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Beispiel 1 – keine zusätzlichen Argumente

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz enthält eine Reihe von Transaktionen für den Monat Januar und wird in eine Tabelle namens `Transactions` geladen. Die `DateFormat`-Standardsystemvariable MM/TT/JJJJ wird verwendet.
- Erstellen Sie ein Feld `in_lunar_week_to_date`, das bestimmt, welche Transaktionen bisher in der Mondwoche des 10. Januar stattfanden.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        inlunarweektodate(date,'01/10/2022', 0) as in_lunar_week_to_date
        ;
Load
*
Inline
[
id,date,amount
8188,'1/10/2022',37.23
8189,'1/17/2022',17.17
8190,'1/26/2022',88.27
8191,'1/12/2022',57.42
8192,'1/19/2022',53.80
8193,'1/21/2022',82.06
8194,'1/1/2022',40.39
8195,'1/27/2022',87.21
8196,'1/11/2022',95.93
8197,'1/29/2022',45.89
8198,'1/31/2022',36.23
8199,'1/18/2022',25.66
8200,'1/23/2022',82.77
8201,'1/15/2022',69.98
8202,'1/4/2022',76.11
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- date
- in_lunar_week_to_date

Ergebnistabelle

date	in_lunar_week_to_date
1/1/2022	0
1/4/2022	0
1/10/2022	-1
1/11/2022	0
1/12/2022	0
1/15/2022	0
1/17/2022	0
1/18/2022	0
1/19/2022	0
1/21/2022	0
1/23/2022	0
1/26/2022	0
1/27/2022	0
1/29/2022	0
1/31/2022	0

Funktion `inlunarweektodate()`, keine weiteren Argumente

Das Feld `in_lunar_week_to_date` wird im vorangehenden load-Befehl erstellt, indem die Funktion `inlunarweektodate()` verwendet und das Feld `date`, ein hartcodierter Zeitstempel für den 10. Januar als `base_date` und ein Versatz von 0 als Argumente der Funktion übergeben werden.

Da Mondwochen am 1. Januar beginnen, fällt der 10. Januar in die Mondwoche, die am 8. Januar beginnt, und weil die Funktion `inlunarweektodate()` verwendet wird, endet diese Mondwoche am 10. Daher gibt jede Transaktion, die zwischen diesen beiden Datumswerten im Januar stattfindet, einen booleschen Wert von `TRUE` zurück. Dies wird in der Ergebnistabelle validiert.

Beispiel 2 – period_no

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript verwendet den gleichen Datensatz und das gleiche Szenario wie das erste Beispiel. In diesem Beispiel besteht die Aufgabe aber darin, ein Feld `2_lunar_weeks_later` zu erstellen, das bestimmt, ob die Transaktionen zwei Wochen nach der laufenden Mondwoche des 1. Januar stattfanden oder nicht.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';
Transactions:
    Load
        *,
        inlunarweektodate(date,'01/10/2022', 2) as [2_lunar_weeks_later]
        ;
Load
*
Inline
[
id,date,amount
8188,'1/10/2022',37.23
8189,'1/17/2022',17.17
8190,'1/26/2022',88.27
8191,'1/12/2022',57.42
8192,'1/19/2022',53.80
8193,'1/21/2022',82.06
8194,'1/1/2022',40.39
8195,'1/27/2022',87.21
8196,'1/11/2022',95.93
8197,'1/29/2022',45.89
8198,'1/31/2022',36.23
8199,'1/18/2022',25.66
8200,'1/23/2022',82.77
8201,'1/15/2022',69.98
8202,'1/4/2022',76.11
];
```

Ergebnisse

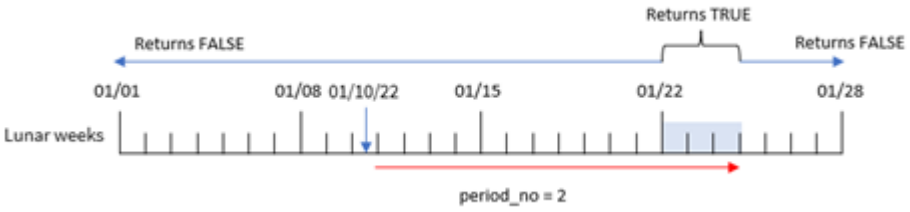
Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `2_lunar_weeks_later`

Ergebnistabelle

date	2_lunar_weeks_later
1/1/2022	0
1/4/2022	0
1/10/2022	0
1/11/2022	0
1/12/2022	0
1/15/2022	0
1/17/2022	0
1/18/2022	0
1/19/2022	0
1/21/2022	0
1/23/2022	-1
1/26/2022	0
1/27/2022	0
1/29/2022	0
1/31/2022	0

Funktion `inlunarweektodate()`, Beispiel „period_no“



In diesem Fall bestimmt die Funktion `inlunarweektodate()`, dass die Mondwoche bis zum 10. Januar drei Tagen entspricht (dem 8., 9. und 10. Januar). Da eine `period_no` von 2 als Versatzargument verwendet wurde, wird diese Mondwoche um 14 Tage verschoben. Somit wird definiert, dass die Dreitage-Mondwoche den 22., 23. und 24. Januar umfasst. Jede Transaktion, die vom 22. Januar bis zum 24. Januar stattfindet, gibt ein boolesches Ergebnis von TRUE zurück.

Beispiel 3 – first_week_day

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Derselbe Datensatz und dasselbe Szenario wie im ersten Beispiel.
- Die `DateFormat`-Standardsystemvariable MM/TT/JJJJ wird verwendet.
- Ein Argument für `first_week_date` von 3. Damit wird der Mondwochenbeginn auf den 3. Januar festgelegt.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        inlunarweek(date,'01/10/2022', 0,3) as in_lunar_week_to_date
        ;
Load
*
Inline
[
id,date,amount
8188,'1/10/2022',37.23
8189,'1/17/2022',17.17
8190,'1/26/2022',88.27
8191,'1/12/2022',57.42
8192,'1/19/2022',53.80
8193,'1/21/2022',82.06
8194,'1/1/2022',40.39
8195,'1/27/2022',87.21
8196,'1/11/2022',95.93
8197,'1/29/2022',45.89
8198,'1/31/2022',36.23
8199,'1/18/2022',25.66
8200,'1/23/2022',82.77
8201,'1/15/2022',69.98
8202,'1/4/2022',76.11
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

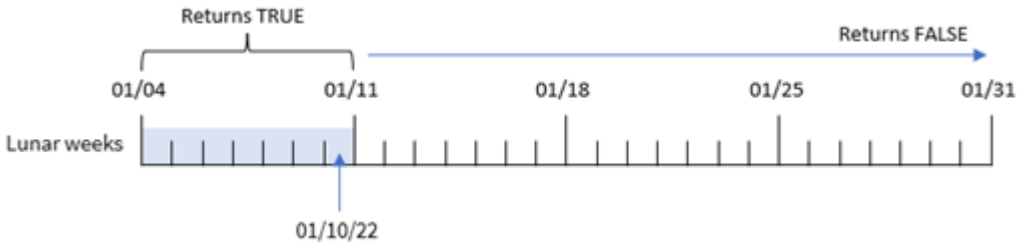
- `date`
- `in_lunar_week_to_date`

Ergebnistabelle

date	in_lunar_week_to_date
1/1/2022	0

date	in_lunar_week_to_date
1/4/2022	-1
1/10/2022	-1
1/11/2022	0
1/12/2022	0
1/15/2022	0
1/17/2022	0
1/18/2022	0
1/19/2022	0
1/21/2022	0
1/23/2022	0
1/26/2022	0
1/27/2022	0
1/29/2022	0
1/31/2022	0

Funktion `inlunarweektodate()`, Beispiel „first_week_day“



Da in diesem Fall das Argument für `the first_week_date` von 3 in der Funktion `inlunarweek()` verwendet wird, dauert die erste Mondwoche vom 3. bis zum 10. Januar. Da der 10. Januar auch das `base_date` ist, gibt jede Transaktion, die zwischen diesen beiden Datumswerten liegt, einen booleschen Wert von `TRUE` zurück.

Beispiel 4 – Diagrammobjektbeispiel

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript verwendet den gleichen Datensatz und das gleiche Szenario wie das erste Beispiel.

In diesem Beispiel wird jedoch der unveränderte Datensatz in die Anwendung geladen. Die Berechnung, die bestimmt, ob Transaktionen in der Mondwoche bis zum 10. Januar stattfanden, wird als Kennzahl in einem Diagrammobjekt der Anwendung erstellt.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8188,'1/10/2022',37.23
8189,'1/17/2022',17.17
8190,'1/26/2022',88.27
8191,'1/12/2022',57.42
8192,'1/19/2022',53.80
8193,'1/21/2022',82.06
8194,'1/1/2022',40.39
8195,'1/27/2022',87.21
8196,'1/11/2022',95.93
8197,'1/29/2022',45.89
8198,'1/31/2022',36.23
8199,'1/18/2022',25.66
8200,'1/23/2022',82.77
8201,'1/15/2022',69.98
8202,'1/4/2022',76.11
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu: `date`.

Erstellen Sie die folgende Kennzahl:

```
=inlunarweektodate(date,'01/10/2022', 0)
```

Ergebnistabelle

date	**=inlunarweektodate(date,'01/10/2022', 0)**
1/1/2022	0
1/4/2022	0
1/10/2022	-1
1/11/2022	0
1/12/2022	0

date	=inlunarweektodate(date,'01/10/2022', 0)
1/15/2022	0
1/17/2022	0
1/18/2022	0
1/19/2022	0
1/21/2022	0
1/23/2022	0
1/26/2022	0
1/27/2022	0
1/29/2022	0
1/31/2022	0

Funktion `inlunarweektodate()`, Diagrammobjektbeispiel

Die Kennzahl `in_lunar_week_to_date` wird im Diagrammobjekt erstellt, indem die Funktion `inlunarweektodate()` verwendet und das Datumsfeld, ein hartcodierter Zeitstempel für den 10. Januar als `base_date` und ein Versatz von 0 als Argumente der Funktion übergeben werden.

Da Mondwochen am 1. Januar beginnen, fällt der 10. Januar in die Mondwoche, die am 8. Januar beginnt. Da zudem die Funktion `inlunarweektodate()` verwendet wird, endet diese Mondwoche am 10. Daher gibt jede Transaktion, die zwischen diesen beiden Datumswerten im Januar stattfindet, einen booleschen Wert von `TRUE` zurück. Dies wird in der Ergebnistabelle validiert.

Beispiel 5 – Szenario

Ladeskript und Diagrammformeln

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Datensatz, der in eine Tabelle namens `Products` geladen wird
- Informationen bestehend aus Produkt-ID, Herstellungsdatum und Selbstkosten

Es wurde festgestellt, dass aufgrund eines Ausrüstungsfehlers die in der Mondwoche vom 12. Januar hergestellten Produkte mangelhaft waren. Das Problem wurde am 13. Januar behoben. Der Endbenutzer möchte ein Diagrammobjekt, das nach Woche den Status der hergestellten Produkte angibt und zeigt, welche „mangelhaft“ und welche „einwandfrei“ waren und was die in dieser Woche gefertigten Produkte gekostet haben.

Ladeskript

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff]';

Products:
Load
*
Inline
[
product_id,manufacture_date,cost_price
8188,'01/02/2022 12:22:06',37.23
8189,'01/05/2022 01:02:30',17.17
8190,'01/06/2022 15:36:20',88.27
8191,'01/08/2022 10:58:35',57.42
8192,'01/09/2022 08:53:32',53.80
8193,'01/10/2022 21:13:01',82.06
8194,'01/11/2022 00:57:13',40.39
8195,'01/12/2022 09:26:02',87.21
8196,'01/13/2022 15:05:09',95.93
8197,'01/14/2022 18:44:57',45.89
8198,'01/15/2022 06:10:46',36.23
8199,'01/16/2022 06:39:27',25.66
8200,'01/17/2022 10:44:16',82.77
8201,'01/18/2022 18:48:17',69.98
8202,'01/26/2022 04:36:03',76.11
8203,'01/27/2022 08:07:49',25.12
8204,'01/28/2022 12:24:29',46.23
8205,'01/30/2022 11:56:56',84.21
8206,'01/30/2022 14:40:19',96.24
8207,'01/31/2022 05:28:21',67.67
];
```

Ergebnisse

Gehen Sie folgendermaßen vor:

1. Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle.
2. Erstellen Sie eine Dimension, um die Wochennamen anzuzeigen:
   `=weekname(manufacture_date)`
3. Erstellen Sie dann eine Dimension, die die Funktion `inlunarweektodate()` verwendet, um zu identifizieren, welche Produkte mangelhaft und welche einwandfrei sind:
   `=if(inlunarweektodate(manufacture_date,makedate(2022,01,12),0),'Defective','Faultless')`
4. Erstellen Sie eine Kennzahl zum Summieren des `cost_price` der Produkte:

```
=sum(cost_price)
```

5. Legen Sie das **Zahlenformat** der Kennzahl auf **Währung** fest.

Ergebnistabelle

=lunarweekname (manufacture_date)	=if(InLunarWeekToDate(manufacture_date,makedate (2022,01,12),0),'Defective','Faultless')	=Sum(cost_ price)
2022/01	Faultless	$142.67
2022/02	Defective	$320.88
2022/02	Faultless	$141.82
2022/03	Faultless	$214.64
2022/04	Faultless	$147.46
2022/05	Faultless	$248.12

Die Funktion `inlunarweektodate()` gibt einen booleschen Wert zurück, wenn sie das Herstellungsdatum der einzelnen Produkte auswertet. Für diejenigen, die einen booleschen Wert von `TRUE` zurückgeben, markiert sie die Produkte als `'Defective'`. Für jedes Produkt, das einen Wert von `FALSE` zurückgibt und somit nicht in der Mondwoche bis zum 12. Januar hergestellt wurde, werden die Produkte als `'Faultless'` markiert.

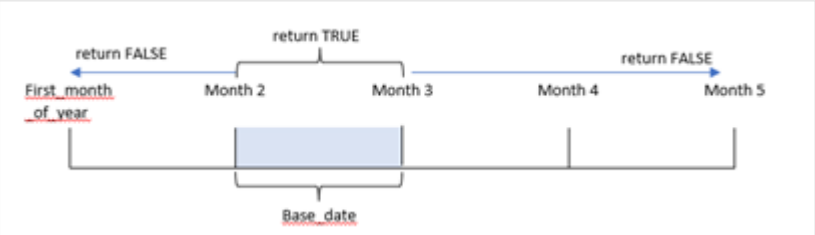
inmonth

Diese Funktion liefert True, wenn **timestamp** innerhalb des Monats liegt, der **base_date** enthält.

Syntax:

```
InMonth (timestamp, base_date, period_no)
```

Diagramm der Funktion `indaytotime`.



Die Funktion `inmonth()` bestimmt also, ob eine Reihe von Datumsangaben innerhalb dieses Monats liegen, und gibt einen booleschen Wert basierend auf einem `base_date` an, das den Monat identifiziert.

Verwendung

Die Funktion `inmonth()` gibt einen booleschen Wert zurück. In der Regel wird dieser Funktionstyp als Bedingung in einer `if expression` verwendet. Dies gibt eine Aggregierung oder Berechnung zurück, abhängig davon, ob ein Datum in den Monat fällt, einschließlich des betreffenden Datums.

Beispielsweise kann die Funktion `inmonth()` verwendet werden, um alle in einem bestimmten Monat gefertigten Geräte zu identifizieren.

Rückgabe Datentyp: Boolesch

In Qlik Sense wird der boolesche Wert „wahr“ durch -1 dargestellt, der Wert „falsch“ durch 0.

Argumente

Argument	Beschreibung
Zeitstempel	Das Datum, das mit `base_date` verglichen werden soll.
base_date	Datum, das für die Interpretation des Monats verwendet wird. Beachten Sie, dass das `base_date` jeder beliebige Tag innerhalb eines Monats sein kann.
period_no	Mit `period_no` kann der Monat verschoben werden. `period_no` ist eine ganze Zahl, wobei 0 für den Monat steht, der `base_date` enthält. Negative Werte von `period_no` stehen für vorangehende, positive Werte für nachfolgende Monate.

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Funktionsbeispiele

Beispiel	Ergebnis
`inmonth ('25/01/2013', '01/01/2013', 0)`	Gibt „wahr“ zurück
`inmonth('25/01/2013', '23/04/2013', 0)`	Gibt „falsch“ zurück
`inmonth ('25/01/2013', '01/01/2013', -1)`	Gibt „falsch“ zurück
`inmonth ('25/12/2012', '17/01/2013', -1)`	Gibt „wahr“ zurück

Beispiel 1 – keine zusätzlichen Argumente

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz, der einen Satz Transaktionen für die erste Hälfte von 2022 enthält.
- Ein vorangehender load-Befehl mit einer zusätzlichen Variablen, „in_month“, die bestimmt, ob Transaktionen im April stattfanden.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
  Load
    *,
    inmonth(date,'04/01/2022', 0) as in_month
    ;
Load
*
Inline
[
id,date,amount
8188,'1/10/2022',37.23
8189,'1/14/2022',17.17
8190,'1/20/2022',88.27
8191,'1/22/2022',57.42
8192,'2/1/2022',53.80
8193,'2/2/2022',82.06
8194,'2/20/2022',40.39
8195,'4/11/2022',87.21
8196,'4/13/2022',95.93
8197,'4/15/2022',45.89
8198,'4/25/2022',36.23
8199,'5/20/2022',25.66
8200,'5/22/2022',82.77
8201,'6/19/2022',69.98
8202,'6/22/2022',76.11
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- date
- in_month

Funktionsbeispiele

date	in_month
1/10/2022	0
1/14/2022	0

date	in_month
1/20/2022	0
1/22/2022	0
2/1/2022	0
2/2/2022	0
2/20/2022	0
4/11/2022	-1
4/13/2022	-1
4/15/2022	-1
4/25/2022	-1
5/20/2022	0
5/22/2022	0
6/19/2022	0
6/22/2022	0

Das Feld „in_month“ wird im vorangehenden load-Befehl erstellt, indem die Funktion `inmonth()` verwendet und das Datumsfeld, ein hartcodierter Zeitstempel für den 1. April als unser `base_date` und eine `period_no` von `0` als Argumente der Funktion übergeben werden.

Das `base_date` identifiziert den Monat, der ein boolesches Ergebnis von TRUE zurückgibt. Daher geben alle Transaktionen, die im April stattfanden, TRUE zurück, was in der Ergebnistabelle validiert wird.

Beispiel 2 – period_no

Ladeskript und Ergebnisse

Übersicht

Es werden derselbe Datensatz und dasselbe Szenario wie im ersten Beispiel verwendet.

In diesem Beispiel erstellen Sie jedoch ein Feld „2_months_prior“, das angibt, ob die Transaktionen zwei Monate vor April stattfanden.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
   *,
   inmonth(date,'04/01/2022', -2) as [2_months_prior]
Inline
[
id,date,amount
8188,'1/10/2022',37.23
```

```
8189,'1/14/2022',17.17
8190,'1/20/2022',88.27
8191,'1/22/2022',57.42
8192,'2/1/2022',53.80
8193,'2/2/2022',82.06
8194,'2/20/2022',40.39
8195,'4/11/2022',87.21
8196,'4/13/2022',95.93
8197,'4/15/2022',45.89
8198,'4/25/2022',36.23
8199,'5/20/2022',25.66
8200,'5/22/2022',82.77
8201,'6/19/2022',69.98
8202,'6/22/2022',76.11
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `2_months_prior`

Funktionsbeispiele

date	2_months_prior
1/10/2022	0
1/14/2022	0
1/20/2022	0
1/22/2022	0
2/1/2022	-1
2/2/2022	-1
2/20/2022	-1
4/11/2022	0
4/13/2022	0
4/15/2022	0
4/25/2022	0
5/20/2022	0
5/22/2022	0
6/19/2022	0
6/22/2022	0

Wenn Sie -2 als Argument für `period_no` in der Funktion `inmonth()` verwenden, wird der vom Argument `base_date` definierte Monat um zwei Monate zurück verschoben. In diesem Beispiel wird der definierte Monat von April zu Februar geändert.

Wenn also eine Transaktion im Februar stattfand, wird ein boolesches Ergebnis von TRUE zurückgegeben.

Beispiel 3 – Diagrammobjekt

Ladeskript und Diagrammformel

Überblick

Es werden derselbe Datensatz und dasselbe Szenario wie in den vorigen Beispielen verwendet.

In diesem Beispiel wird jedoch der unveränderte Datensatz in die Anwendung geladen. Die Berechnung, die bestimmt, ob Transaktionen im April stattfanden, wird als Kennzahl in einem Diagrammobjekt der Anwendung erstellt.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8188,'1/10/2022',37.23
8189,'1/14/2022',17.17
8190,'1/20/2022',88.27
8191,'1/22/2022',57.42
8192,'2/1/2022',53.80
8193,'2/2/2022',82.06
8194,'2/20/2022',40.39
8195,'4/11/2022',87.21
8196,'4/13/2022',95.93
8197,'4/15/2022',45.89
8198,'4/25/2022',36.23
8199,'5/20/2022',25.66
8200,'5/22/2022',82.77
8201,'6/19/2022',69.98
8202,'6/22/2022',76.11
];
```

Diagrammobjekt

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu:

`date`
Um zu berechnen, ob eine Transaktion im April stattfand, erstellen Sie die folgende Kennzahl:

```
=inmonth(date,'04/01/2022', 0)
```

Ergebnisse

	Funktionsbeispiele
date	**=inmonth(date,'04/01/2022', 0)**
1/10/2022	0
1/14/2022	0
1/20/2022	0
1/22/2022	0
2/1/2022	0
2/2/2022	0
2/20/2022	0
4/11/2022	-1
4/13/2022	-1
4/15/2022	-1
4/25/2022	-1
5/20/2022	0
5/22/2022	0
6/19/2022	0
6/22/2022	0

Beispiel 4 – Szenario

Ladeskript und Ergebnisse

Übersicht

In diesem Beispiel wird ein Datensatz in eine Tabelle namens „`Products`“ geladen. Die Tabelle enthält die folgenden Felder:

- Produkt-ID
- Herstellungsdatum
- Selbstkosten

Aufgrund eines Maschinenfehlers waren die im Monat Juli 2022 gefertigten Produkte mangelhaft. Das Problem wurde am 27. Juli 2022 behoben.

Der Endbenutzer möchte ein Diagramm haben, das nach Monat den Status der hergestellten Produkte angibt und zeigt, welche „mangelhaft“ (boolesch TRUE) und welche „einwandfrei“ (boolesch FALSE) waren und was die in diesem Monat gefertigten Produkte gekostet haben.

Ladeskript

```
Products:
Load
*
Inline
[
product_id,manufacture_date,cost_price
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu:

`=monthname(manufacture_date)`
Erstellen Sie die folgenden Kennzahlen:

- `=sum(cost_price)`
- `=if(only(inmonth(manufacture_date,makedate(2022,07,01),0)),'Defective','Faultless')`

1. Legen Sie das **Zahlenformat** der Kennzahl auf **Währung** fest.
2. Deaktivieren Sie unter **Darstellung** die Option **Gesamtwerte**.

Ergebnistabelle

monthname (manufacture_date)	=if(only(inmonth(manufacture_date,makedate (2022,07,01),0)),'Defective','Faultless')	sum(cost_ price)
Jan 2022	Faultless	$54.40
Feb 2022	Faultless	$145.69

monthname (manufacture_date)	=if(only(inmonth(manufacture_date,makedate (2022,07,01),0)),'Defective','Faultless')	sum(cost_ price)
Mar 2022	Faultless	$53.80
Apr 2022	Faultless	$82.06
May 2022	Faultless	$127.60
Jun 2022	Faultless	$141.82
Jul 2022	Defective	$214.64
Aug 2022	Faultless	$147.46
Sep 2022	Faultless	$84.21
Oct 2022	Faultless	$163.91

Die Funktion `inmonth()` gibt einen booleschen Wert zurück, wenn sie das Herstellungsdatum der einzelnen Produkte auswertet. Für alle im Juli 2022 hergestellten Produkte gibt die Funktion `inmonth()` einen booleschen Wert von TRUE zurück und markiert die Produkte als „Mangelhaft“. Alle Produkte, die einen Wert von FALSE zurückgeben und daher nicht im Juli gefertigt wurden, werden als „Einwandfrei“ markiert.

inmonths

Diese Funktion ermittelt, ob ein Zeitstempel im gleichen Monat, Zweimonatszeitraum, Quartal, Viermonatszeitraum oder Halbjahr wie ein Basisdatum liegt. Es lässt sich auch bestimmen, ob ein Zeitstempel in den vorhergehenden oder nachfolgenden Zeitraum fällt.

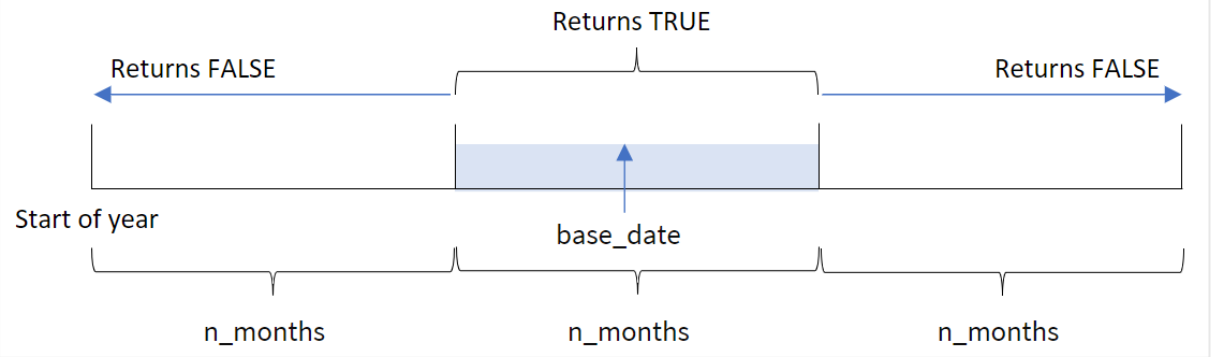
Syntax:

```
InMonths(n_months, timestamp, base_date, period_no [, first_month_of_year])
```

Rückgabe Datentyp: Boolesch

In Qlik Sense wird der boolesche Wert „wahr“ durch -1 dargestellt, der Wert „falsch“ durch 0.

Diagramm der Funktion `inmonths()`



Die Funktion `inmonths()` unterteilt das Jahr in Segmente, gestützt auf das angegebene Argument `n_months`. Dann bestimmt sie, ob die einzelnen ausgewerteten Zeitstempel in das gleiche Segment wie das Argument `base_date` fallen. Wenn aber das Argument `period_no` angegeben wird, bestimmt die Funktion, ob die Zeitstempel in einen Zeitraum vor oder nach dem `base_date` fallen.

Die folgenden Segmente des Jahres sind in der Funktion als Argumente `n_month` verfügbar.

Argumente `n_month`

Zeitraum	Anzahl der Monate
Monat	1
Zweimonatszeitraum	2
Quartal	3
Viermonatszeitraum	4
Halbjahr	6

Verwendung

Die Funktion `inmonths()` gibt einen booleschen Wert zurück. In der Regel wird dieser Funktionstyp als Bedingung in einer `if expression` verwendet. Wenn Sie die Funktion `inmonths()` verwenden, können Sie den Zeitraum auswählen, den Sie auswerten möchten. Sie können dem Benutzer beispielsweise die Möglichkeit geben, Produkte zu identifizieren, die im Monat, Quartal oder Halbjahr eines bestimmten Zeitraums hergestellt wurden.

Rückgabe Datentyp: Boolesch

In Qlik Sense wird der boolesche Wert „wahr“ durch -1 dargestellt, der Wert „falsch“ durch 0.

Argumente

Argument	Beschreibung
n_months	Die Anzahl der Monate, die den Zeitraum definiert. Eine Ganzzahl oder eine Formel, die eine Ganzzahl mit einem der folgenden Werte ergibt: 1 (entspricht der Funktion inmonth()), 2 (Zweimonatszeitraum), 3 (entspricht der Funktion inquarter()), 4 (Viermonatszeitraum) oder 6 (Halbjahr).
timestamp	Das Datum, das mit **base_date** verglichen werden soll.
base_date	Datum, das für die Interpretation des Zeitraums verwendet wird.
period_no	Mit **period_no**, einer ganze Zahl oder einer Formel, die eine ganze Zahl ergibt, kann ein anderer Beginn für den Zeitraum festgelegt werden, wobei 0 für den Zeitraum steht, der **base_date** enthält. Negative Werte von **period_no** stehen für vorangehende, positive Werte für nachfolgende Zeiträume.
first_month_of_year	Wenn Sie mit abweichenden Geschäftsjahren arbeiten möchten, definieren Sie mit einer Zahl zwischen 2 und 12 für **first_month_of_year** einen anderen Startmonat.

Sie können die folgenden Werte verwenden, um den ersten Monat des Jahres im Argument `first_month_of_year` festzulegen.

Werte für first_month_of_year

Monat	Wert
Februar	2
März	3
April	4
Mai	5
Juni	6
Juli	7
August	8
September	9
Oktober	10
November	11
Dezember	12

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Funktionsbeispiele

Beispiel	Ergebnis
`inmonths(4, '01/25/2013', '04/25/2013', 0)`	Gibt TRUE zurück. Weil der Wert von timestamp, 01/25/2013, in einen Viermonatszeitraum von 01/01/2013 bis 04/30/2013 fällt, in dem der Wert von `base_date`, 04/25/2013 liegt.

Beispiel	Ergebnis
`inmonths(4, '05/25/2013', '04/25/2013', 0)`	Gibt FALSE zurück. Weil 05/25/2013 sich außerhalb des im vorhergehenden Beispiel angegebenen Zeitraums befindet.
`inmonths(4, '11/25/2012', '02/01/2013', -1)`	Gibt TRUE zurück. Weil der Wert von `period_no`, `-1`, den Suchzeitraum um einen Zeitraum von vier Monaten (den Wert von n-months) nach hinten verschiebt. Dadurch ergibt sich der Suchzeitraum 09/01/2012 bis 12/31/2012.
`inmonths(4, '05/25/2006', '03/01/2006', 0, 3)`	Gibt TRUE zurück. Da der Wert von `first_month_of_year` auf `3` festgelegt ist, wodurch der Suchzeitraum 03/01/2006 bis 07/30/2006 anstelle von 01/01/2006 bis 04/30/2006 ist.

Beispiel 1 – keine zusätzlichen Argumente

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz, der eine Reihe von Transaktionen für 2022 enthält, wird in eine Tabelle namens „`Transactions`“ geladen.
- Ein vorangehender load-Befehl mit einer zusätzlichen Variablen „`in_months`“ bestimmt, ob Transaktionen im gleichen Quartal wie der 15. Mai 2022 stattfanden.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
       *,
       inmonths(3,date,'05/15/2022', 0) as in_months
       ;
Load
*
Inline
[
id,date,amount
8188,'2/19/2022',37.23
8189,'3/7/2022',17.17
8190,'3/30/2022',88.27
8191,'4/5/2022',57.42
8192,'4/16/2022',53.80
8193,'5/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/22/2022',87.21
```

```
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `in_months`

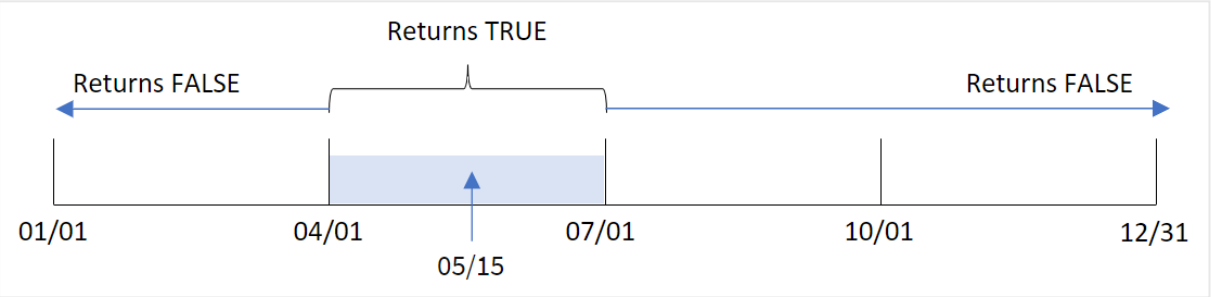
Ergebnistabelle

date	in_months
2/19/2022	0
3/7/2022	0
3/30/2022	0
4/5/2022	-1
4/16/2022	-1
5/1/2022	-1
5/7/2022	-1
5/22/2022	-1
6/15/2022	-1
6/26/2022	-1
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0

date	in_months
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Das Feld „in_months“ wird im vorangehenden load-Befehl mithilfe der Funktion `inmonths()` erstellt. Das erste angegebene Argument ist `3`. Damit wird das Jahr in Quartalssegmente unterteilt. Das zweite Argument identifiziert, welches Feld ausgewertet wird. In diesem Beispiel ist dies das Datumsfeld. Das dritte Argument ist ein hartcodiertes Datum für den 15. Mai, das `base_date`, und eine `period_no` von `0` ist das letzte Argument.

Diagramm der Funktion `inmonths()` mit Quartalssegmenten



Der Monat Mai liegt im zweiten Quartal des Jahres. Daher gibt jede Transaktion, die zwischen dem 1. April und dem 30. Juni stattfindet, ein boolesches Ergebnis von TRUE zurück. Dies wird in der Ergebnistabelle validiert.

Beispiel 2 – period_no

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz, der eine Reihe von Transaktionen für 2022 enthält, wird in eine Tabelle namens „`Transactions`“ geladen.
- Eine vorangehende load-Anweisung mit einer zusätzlichen Variablen, „`previous_quarter`“, die bestimmt, ob Transaktionen im Quartal vor dem 15. Mai 2022 stattfanden.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
       *,
       inmonths(3,date,'05/15/2022', -1) as previous_quarter
```

```
      ;
Load
*
Inline
[
id,date,amount
8188,'2/19/2022',37.23
8189,'3/7/2022',17.17
8190,'3/30/2022',88.27
8191,'4/5/2022',57.42
8192,'4/16/2022',53.80
8193,'5/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/22/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `previous_quarter`

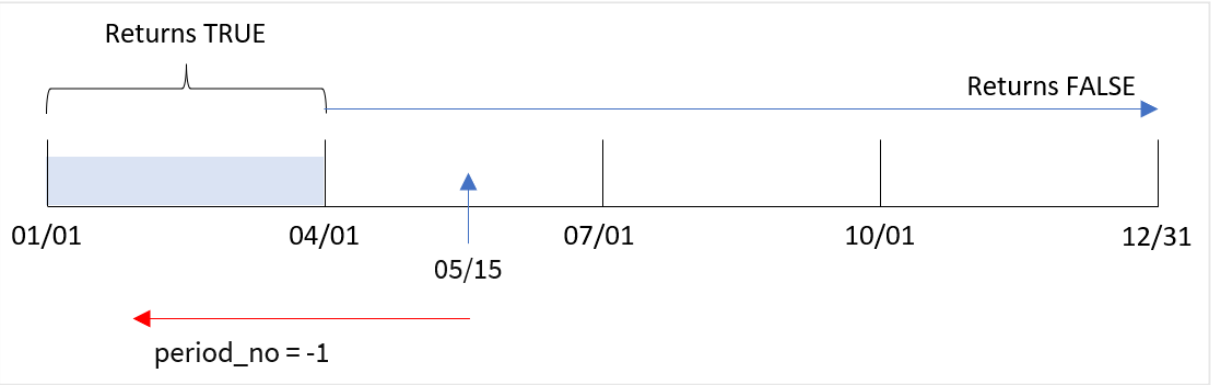
Ergebnistabelle

date	**voriges Quartal**
2/19/2022	-1
3/7/2022	-1
3/30/2022	-1
4/5/2022	0
4/16/2022	0
5/1/2022	0
5/7/2022	0
5/22/2022	0

date	voriges Quartal
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Die Funktion wertet aus, ob Transaktionen im ersten Quartal des Jahres stattfanden. Dafür wird `-1` als Argument `period_no` in der Funktion `inmonths()` verwendet. Der 15. Mai ist das `base_date` und liegt im zweiten Quartal des Jahres (April-Juni).

Diagramm der Funktion `inmonths()` mit Quartalssegmenten, wobei `period_no` auf `-1` festgelegt ist



Daher gibt jede Transaktion, die zwischen Januar und März stattfindet, ein boolesches Ergebnis von TRUE zurück.

Beispiel 3 – first_month_of_year

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz, der eine Reihe von Transaktionen für 2022 enthält, wird in eine Tabelle namens „Transactions“ geladen.
- Eine vorangehende load-Anweisung mit einer zusätzlichen Variablen „in_months“ bestimmt, ob Transaktionen im gleichen Quartal wie der 15. Mai 2022 stattfanden.

In diesem Beispiel legt die Organisationsrichtlinie fest, dass März der erste Monat des Geschäftsjahres ist.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
       *,
       inmonths(3,date,'05/15/2022', 0, 3) as in_months
       ;
Load
*
Inline
[
id,date,amount
8188,'2/19/2022',37.23
8189,'3/7/2022',17.17
8190,'3/30/2022',88.27
8191,'4/5/2022',57.42
8192,'4/16/2022',53.80
8193,'5/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/22/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

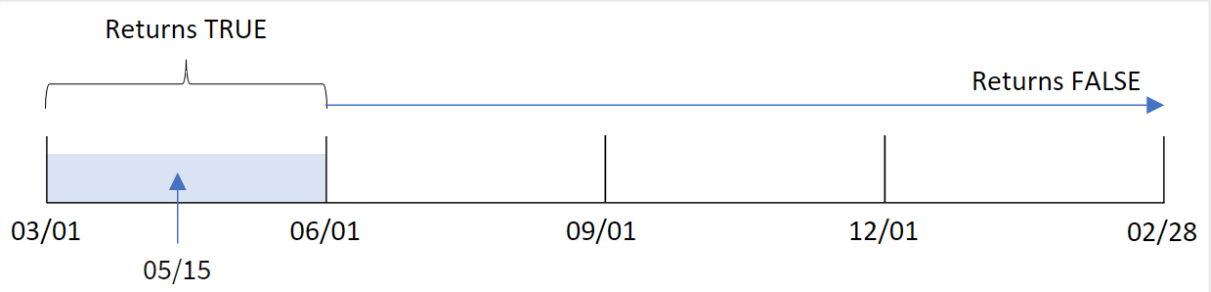
- date
- in_months

Ergebnistabelle

date	in_months
2/19/2022	0
3/7/2022	-1
3/30/2022	-1
4/5/2022	-1
4/16/2022	-1
5/1/2022	-1
5/7/2022	-1
5/22/2022	-1
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Da `3` als Argument `first_month_of_year` in der Funktion `inmonths()` verwendet wird, beginnt die Funktion das Jahr am 1. März. Die Funktion `inmonths()` unterteilt dann das Jahr in Quartale: Mär-Mai, Jun-Aug, Sep-Nov, Dez-Feb. Somit fällt der 15. Mai in das erste Quartal des Jahres (Mär-Mai).

Diagramm der Funktion `inmonths()` mit März als erstem Monat des Jahres



Jede Transaktion, die in diesen Monaten stattfindet, gibt ein boolesches Ergebnis von TRUE zurück.

Beispiel 4 – Diagrammobjektbeispiel

Ladeskript und Diagrammformel

Übersicht

Es werden derselbe Datensatz und dasselbe Szenario wie im ersten Beispiel verwendet.

In diesem Beispiel wird jedoch der unveränderte Datensatz in die Anwendung geladen. Die Berechnung, die bestimmt, ob Transaktionen im gleichen Quartal wie der 15. Mai 2022 stattfanden, wird als Kennzahl in einem Diagramm in der App erstellt.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8188,'2/19/2022',37.23
8189,'3/7/2022',17.17
8190,'3/30/2022',88.27
8191,'4/5/2022',57.42
8192,'4/16/2022',53.80
8193,'5/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/22/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu:

- `date`

Um zu berechnen, ob Transaktionen im selben Quartal wie der 15. Mai stattfanden, erstellen Sie die folgende Kennzahl:

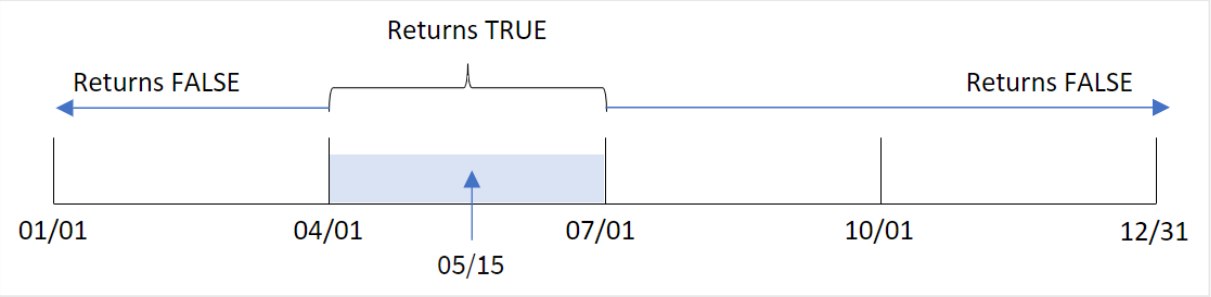
```
=inmonths(3,date,'05/15/2022', 0)
```

Ergebnistabelle

date	=inmonths(3,date,'05/15/2022', 0)
2/19/2022	0
3/7/2022	0
3/30/2022	0
4/5/2022	-1
4/16/2022	-1
5/1/2022	-1
5/7/2022	-1
5/22/2022	-1
6/15/2022	-1
6/26/2022	-1
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Das Feld „`in_months`“ wird im Diagramm mithilfe der Funktion `inmonths()` erstellt. Das erste angegebene Argument ist `3`. Damit wird das Jahr in Quartalssegmente unterteilt. Das zweite Argument identifiziert, welches Feld ausgewertet wird. In diesem Beispiel ist dies das Datumsfeld. Das dritte Argument ist ein hartcodiertes Datum für den 15. Mai, das `base_date`, und eine `period_no` von `0` ist das letzte Argument.

Diagramm der Funktion `inmonths()` mit Quartalssegmenten



Der Monat Mai liegt im zweiten Quartal des Jahres. Daher gibt jede Transaktion, die zwischen dem 1. April und dem 30. Juni stattfindet, ein boolesches Ergebnis von TRUE zurück. Dies wird in der Ergebnistabelle validiert.

Beispiel 5 – Szenario

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Datensatz, der in eine Tabelle namens „`Products`“ geladen wird
- Die Tabelle enthält die folgenden Felder:
    - product ID
    - product type
    - manufacture date
    - cost price

Der Endbenutzer möchte ein Diagramm, das nach Produkttyp die Kosten der im ersten Segment des Jahres 2021 gefertigten Produkte anzeigt. Der Benutzer möchte die Länge dieses Segments definieren können.

Ladeskript

```
SET vPeriod = 1;

Products:
Load
*
Inline
[
product_id,product_type,manufacture_date,cost_price
8188,product A,'2/19/2022',37.23
8189,product D,'3/7/2022',17.17
8190,product C,'3/30/2022',88.27
8191,product B,'4/5/2022',57.42
8192,product D,'4/16/2022',53.80
8193,product D,'5/1/2022',82.06
8194,product A,'5/7/2022',40.39
```

```
8195,product B,'5/22/2022',87.21
8196,product C,'6/15/2022',95.93
8197,product B,'6/26/2022',45.89
8198,product C,'7/9/2022',36.23
8199,product D,'7/22/2022',25.66
8200,product D,'7/23/2022',82.77
8201,product A,'7/27/2022',69.98
8202,product A,'8/2/2022',76.11
8203,product B,'8/8/2022',25.12
8204,product B,'8/19/2022',46.23
8205,product B,'9/26/2022',84.21
8206,product C,'10/14/2022',96.24
8207,product D,'10/29/2022',67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt.

Am Beginn des Ladeskripts wurde eine Variable (`vPeriod`) erstellt, die an die Variableneingabesteuerung gebunden wird.

Gehen Sie folgendermaßen vor:

1. Klicken Sie im Extras-Fenster auf **Benutzerdefinierte Objekte**.
2. Wählen Sie **Qlik Dashboard Bundle** aus und erstellen Sie ein Objekt **Variableneingabe**.
3. Geben Sie einen Titel für das Diagrammobjekt ein.
4. Wählen Sie unter **Variable** den Eintrag **vPeriod** als den Namen aus und legen Sie das Objekt so fest, dass es als **Dropdown** angezeigt wird.
5. Klicken Sie unter **Werte** auf **Dynamisch**. Geben Sie Folgendes ein:
   `='1~month|2~bi-month|3~quarter|4~tertial|6~half-year'.`
6. Fügen Sie eine neue Tabelle zum Arbeitsblatt hinzu.
7. Fügen Sie im Eigenschaftsfenster unter **Daten** das Feld `product_type` als Dimension hinzu.
8. Fügen Sie die folgende Formel als Kennzahl hinzu:
   `=sum(if(inmonths($(vPeriod),manufacture_date,makedate(2022,01,01),0),cost_price,0))`
9. Legen Sie das **Zahlenformat** der Kennzahl auf **Währung** fest.

Ergebnistabelle

product_type	=sum(if(inmonths($(vPeriod),manufacture_date,makedate(2022,01,01),0),cost_price,0))
Produkt A	$88.27
Produkt B	$37.23
Produkt C	$17.17
Produkt D	$0.00

Die Funktion `inmonths()` verwendet die Benutzereingabe als Argument zum Definieren der Größe des Startsegments des Jahres. Die Funktion übergibt das Herstellungsdatum der einzelnen Produkte als das zweite Argument der Funktion `inmonths()`. Da der 1. Januar als drittes Argument der Funktion `inmonths()` verwendet wird, geben Produkte mit einem Herstellungsdatum, das in das Anfangssegment des Jahres fällt, einen booleschen Wert von TRUE zurück, und daher addiert die Summenfunktion die Kosten dieser Produkte.

inmonthstodate

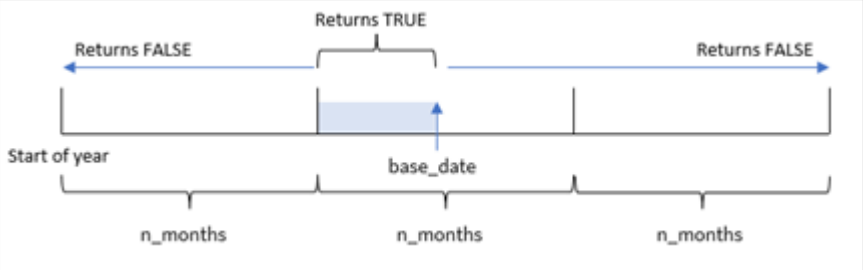
Diese Funktion ermittelt, ob ein Zeitstempel im Teil des Zeitraums von einem Monat, Zweimonatszeitraum, Quartal, Viermonatszeitraum oder Halbjahr liegt, bis einschließlich zur letzten Millisekunde von `base_date`. Es lässt sich auch bestimmen, ob ein Zeitstempel in den vorhergehenden oder nachfolgenden Zeitraum fällt.

Syntax:

```
InMonths (n_months, timestamp, base_date, period_no[, first_month_of_year ])
```

Rückgabe Datentyp: Boolesch

Diagramm der Funktion `inmonthstodate`.



Argumente

Argument	Beschreibung
n_months	Die Anzahl der Monate, die den Zeitraum definiert. Eine Ganzzahl oder eine Formel, die eine Ganzzahl mit einem der folgenden Werte ergibt: 1 (entspricht der Funktion inmonth ()), 2 (Zweimonatszeitraum), 3 (entspricht der Funktion inquarter()), 4 (Viermonatszeitraum) oder 6 (Halbjahr).
timestamp	Das Datum, das mit **base_date** verglichen werden soll.
base_date	Datum, das für die Interpretation des Zeitraums verwendet wird.
period_no	Mit **period_no**, einer ganze Zahl oder einer Formel, die eine ganze Zahl ergibt, kann ein anderer Beginn für den Zeitraum festgelegt werden, wobei 0 für den Zeitraum steht, der **base_date** enthält. Negative Werte von **period_no** stehen für vorangehende, positive Werte für nachfolgende Zeiträume.
first_month_of_year	Wenn Sie mit abweichenden Geschäftsjahren arbeiten möchten, definieren Sie mit einer Zahl zwischen 2 und 12 für **first_month_of_year** einen anderen Startmonat.

In der Funktion `inmonthstodate()` fungiert das `base_date` als Endpunkt des bestimmten Jahressegments, zu dem es gehört.

Wenn beispielsweise das Jahr in Viermonatssegmente unterteilt wurde und das `base_date` der 15. Mai war, gibt jeder Zeitstempel zwischen Anfang Januar und Ende April ein boolesches Ergebnis von FALSE zurück. Datumswerte zwischen dem 1. und dem 15. Mai geben TRUE zurück. Der Rest des Jahres gibt wiederum FALSE zurück.

Diagramm des booleschen Ergebnisbereichs der Funktion `inmonthstodate`.

Die folgenden Segmente des Jahres sind in der Funktion als Argumente `n_month` verfügbar.

Argumente `n_month`

Zeitraum	Anzahl der Monate
Monat	1
Zweimonatszeitraum	2
Quartal	3
vier Monate	4
Halbjahr	6

Verwendung

Die Funktion `inmonthstodate()` gibt einen booleschen Wert zurück. In der Regel wird dieser Funktionstyp als Bedingung in einer `if expression` verwendet. Wenn Sie die Funktion `inmonthstodate()` verwenden, können Sie den Zeitraum auswählen, den Sie auswerten möchten. Sie können beispielsweise eine Eingabevariable angeben, damit der Benutzer Produkte identifizieren kann, die im Monat, Quartal oder Halbjahr eines Zeitraums bis zu einem bestimmten Datum hergestellt wurden.

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für

Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Funktionsbeispiele

Beispiel	Ergebnis
`inmonthstodate(4, '01/25/2013', '04/25/2013', 0)`	Gibt True zurück, weil der Wert von timestamp (01/25/2013) in dem Viermonatszeitraum 01/01/2013 bis zum Ende von 04/25/2013 liegt, in dem der Wert von base_date (04/25/2013) liegt.
`inmonthstodate(4, '04/26/2013', '04/25/2006', 0)`	Gibt False zurück, weil 04/26/2013 sich außerhalb des im vorhergehenden Beispiel angegebenen Zeitraums befindet.
`inmonthstodate(4, '09/25/2005', '02/01/2006', -1)`	Gibt True zurück, weil der Wert von period_no (-1) den Suchzeitraum um einen Zeitraum von vier Monaten (den Wert von n-months) nach hinten verschiebt. Dadurch ergibt sich der Suchzeitraum 01/09/2005 bis 02/01/2006.
`inmonthstodate(4, '04/25/2006', '06/01/2006', 0, 3)`	Gibt True zurück, weil der Wert von first_month_of_year auf 3 festgelegt ist, weshalb der Suchzeitraum 03/01/2006 bis 06/01/2006 anstelle von 05/01/2006 bis 06/01/2006 wird.

Beispiel 1 – keine zusätzlichen Argumente

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Datensatz mit einer Reihe von Transaktionen für 2022, der in eine Tabelle namens „`Transactions`“ geladen wird
- Datumsfeld im Format `(MM/DD/YYYY)` der Systemvariablen `DateFormat`
- Ein vorangehender load-Befehl, der Folgendes enthält:
    - Die Funktion `inmonthstodate()`, die als das Feld „`in_months_to_date`“ festgelegt ist. Dies bestimmt, welche Transaktionen im Quartal bis zum 15. Mai 2022 stattgefunden haben.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
    *,
    inmonthstodate(3,date,'05/15/2022', 0) as in_months_to_date
    ;
Load
```

```
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `in_months_to_date`

Ergebnistabelle

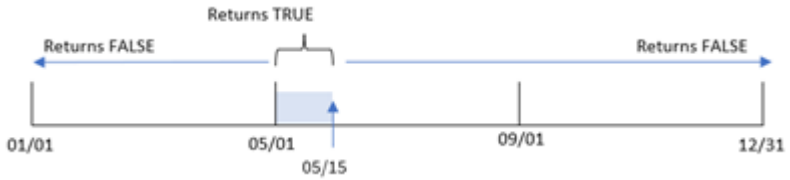
date	in_months_to_date
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	0
4/1/2022	-1
5/7/2022	-1
5/16/2022	0
6/15/2022	0

date	in_months_to_date
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Das Feld „`in_months_to_date`“ wird in der vorangehenden load-Anweisung mithilfe der Funktion `inmonthstodate()` erstellt.

Das erste angegebene Argument ist `3`. Damit wird das Jahr in Quartalssegmente unterteilt. Das zweite Argument identifiziert, welches Feld ausgewertet wird. Das dritte Argument ist ein hartcodiertes Datum, der 15. Mai. Dies ist das `base_date`, das die Endbegrenzung dieses Segments definiert. Eine `period_no` von `0` ist das abschließende Argument.

Diagramm der Funktion `inmonthstodate` ohne zusätzlichen Argumente.



Jede Transaktion, die zwischen dem 1. April und dem 15. Mai stattfindet, gibt ein boolesches Ergebnis von TRUE zurück. Transaktionsdatumswerte außerhalb dieses Zeitraums geben FALSE zurück.

Beispiel 2 – period_no

Ladeskript und Ergebnisse

Übersicht

Es werden derselbe Datensatz und dasselbe Szenario wie im ersten Beispiel verwendet.

In diesem Beispiel besteht die Aufgabe aber darin, ein Feld „`previous_qtr_to_date`“ zu erstellen, das bestimmt, ob die Transaktionen in einem Quartal vor dem 15. Mai stattfanden.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
    *,
    inmonthstodate(3,date,'05/15/2022', -1) as previous_qtr_to_date
    ;
Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `previous_qtr_to_date`

Ergebnistabelle

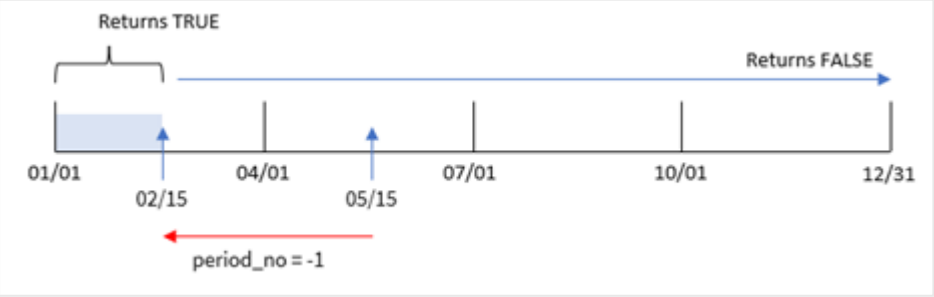
date	previous_qtr_to_date
1/7/2022	-1
1/19/2022	-1
2/5/2022	-1

date	previous_qtr_to_date
2/28/2022	0
3/16/2022	0
4/1/2022	0
5/7/2022	0
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Da `-1` als das Argument `period_no` in der Funktion `inmonthstodate()` verwendet wird, verschiebt die Funktion die Grenzen des Vergleichsjahressegments um ein Quartal.

Der 15. Mai fällt in das zweite Quartal des Jahres, sodass das Segment anfänglich dem Zeitraum vom 1. April bis zum 15. Mai entspricht. Das Argument `period_no` versetzt dieses Segment um negative drei Monate. Die Datumsgrenzen sind jetzt der 1. Januar bis zum 15. Februar.

Diagramm der Funktion `inmonthstodate`, wobei „period_no“ auf -1 festgelegt ist.



Daher gibt jede Transaktion, die zwischen dem 1. Januar und 15. Februar stattfindet, ein boolesches Ergebnis von TRUE zurück.

Beispiel 3 – first_month_of_year

Ladeskript und Ergebnisse

Übersicht

Es werden derselbe Datensatz und dasselbe Szenario wie im ersten Beispiel verwendet.

In diesem Beispiel legt die Organisationsrichtlinie fest, dass März der erste Monat des Geschäftsjahres ist.

Erstellen Sie ein Feld „in_months_to_date“, das bestimmt, welche Transaktionen im gleichen Quartal bis zum 15. Mai 2022 stattgefunden haben.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
    *,
    inmonthstodate(3,date,'05/15/2022', 0,3) as in_months_to_date
    ;
Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `in_months_to_date`

Ergebnistabelle

date	previous_qtr_to_date
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	-1
4/1/2022	-1
5/7/2022	-1
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Da `3` als Argument `first_month_of_year` in der Funktion `inmonthstodate()` verwendet wird, beginnt die Funktion das Jahr am 1. März und unterteilt dann das Jahr basierend auf dem ersten angegebenen Argument in Quartale. Daher lauten die Quartalssegmente wie folgt:

- Mär-Mai
- Jun-Aug

- Sep-Nov
- Dez-Feb

Das base_date vom 15. Mai segmentiert dann das Quartal Mär-Mai, indem seine Endgrenze als 15. Mai festgelegt wird.

Diagramm der Funktion `inmonthstodate` mit März als erstem Monat des Jahres.

Hier gibt jede Transaktion, die zwischen dem 1. März und dem 15. Mai stattfindet, ein boolesches Ergebnis von TRUE zurück, während Transaktionen mit Datum außerhalb dieser Grenzen einen Wert von FALSE zurückgeben.

Beispiel 4 – Diagrammbeispiel

Ladeskript und Diagrammformel

Übersicht

Es werden derselbe Datensatz und dasselbe Szenario wie im ersten Beispiel verwendet.

In diesem Beispiel wird der unveränderte Datensatz in die App geladen. Die Aufgabe besteht im Erstellen einer Berechnung, die bestimmt, ob Transaktionen im gleichen Quartal wie der 15. Mai stattfanden. Sie wird als Kennzahl in einem Diagramm in der App erstellt.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
```

```
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu:

```
date
```

Um zu berechnen, ob Transaktionen im selben Quartal wie der 15. Mai stattfanden, erstellen Sie die folgende Kennzahl:

```
=inmonthstodate(3,date,'05/15/2022', 0)
```

Ergebnistabelle

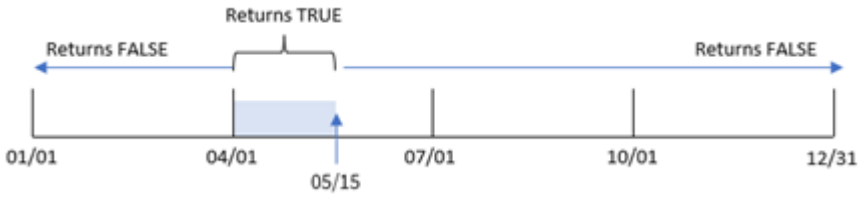
date	=inmonthstodate(3,date,'05/15/2022', 0)
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	0
4/1/2022	-1
5/7/2022	-1
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0

date	=inmonthstodate(3,date,'05/15/2022', 0)
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Die Kennzahl „`'in_months_to_date`“ wird im Diagramm anhand der Funktion `inmonthstodate()` erstellt.

Das erste angegebene Argument ist `3`. Damit wird das Jahr in Quartalssegmente unterteilt. Das zweite Argument identifiziert, welches Feld ausgewertet wird. Das dritte Argument ist ein hartcodiertes Datum, der 15. Mai. Dies ist das `base_date`, das die Endbegrenzung dieses Segments definiert. Eine `period_no` von `0` ist das abschließende Argument.

Diagramm der Funktion `inmonthstodate` mit Quartalssegmenten.



Jede Transaktion, die zwischen dem 1. April und dem 15. Mai stattfindet, gibt ein boolesches Ergebnis von TRUE zurück. Transaktionsdatumswerte außerhalb dieses Segments geben FALSE zurück.

Beispiel 5 – Szenario

Ladeskript und Ergebnisse

Übersicht

In diesem Beispiel wird ein Datensatz in eine Tabelle namens „`Sales`“ geladen. Die Tabelle enthält die folgenden Felder:

- Produkt-ID
- Produkttyp
- Verkaufsdatum
- Verkaufspreis

Der Endbenutzer möchte ein Diagramm, das nach Produkttyp den Umsatz von Produkten zeigt, die im Zeitraum bis zum 24. Dezember 2022 verkauft werden. Der Benutzer möchte die Länge dieses Zeitraums definieren können.

Ladeskript

```
SET vPeriod = 1;

Products:
Load
*
Inline
[
product_id,product_type,sales_date,sales_price
8188,product A,'9/19/2022',37.23
8189,product D,'10/27/2022',17.17
8190,product C,'10/30/2022',88.27
8191,product B,'10/31/2022',57.42
8192,product D,'11/16/2022',53.80
8193,product D,'11/28/2022',82.06
8194,product A,'12/2/2022',40.39
8195,product B,'12/5/2022',87.21
8196,product C,'12/15/2022',95.93
8197,product B,'12/16/2022',45.89
8198,product C,'12/19/2022',36.23
8199,product D,'12/22/2022',25.66
8200,product D,'12/23/2022',82.77
8201,product A,'12/24/2022',69.98
8202,product A,'12/24/2022',76.11
8203,product B,'12/26/2022',25.12
8204,product B,'12/27/2022',46.23
8205,product B,'12/27/2022',84.21
8206,product C,'12/28/2022',96.24
8207,product D,'12/29/2022',67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt.

Am Beginn des Ladeskripts wurde eine Variable (`vPeriod`) erstellt, die an die Variableneingabesteuerung gebunden wird.

Gehen Sie folgendermaßen vor:

1. Klicken Sie im Extras-Fenster auf **Benutzerdefinierte Objekte**.
2. Wählen Sie das **Qlik Dashboard Bundle** aus und fügen Sie eine **Variableneingabe** zu Ihrem Arbeitsblatt hinzu.
3. Geben Sie einen Titel für das Diagramm ein.
4. Wählen Sie unter **Variable** den Eintrag **vPeriod** als den Namen aus und legen Sie das Objekt so fest, dass es als **Dropdown** angezeigt wird.
5. Klicken Sie unter **Werte** auf **Dynamisch**. Geben Sie Folgendes ein:
   `='1~month|2~bi-month|3~quarter|4~tertial|6~half-year'.`
6. Fügen Sie eine neue Tabelle zum Arbeitsblatt hinzu.

7. Fügen Sie im Eigenschaftsfenster unter **Daten** das Feld `product_type` als Dimension hinzu.
8. Fügen Sie die folgende Formel als Kennzahl hinzu:
   `=sum(if(inmonthstodate($(vPeriod),sales_date,makedate(2022,12,24),0),sales_price,0))`
9. Legen Sie das **Zahlenformat** der Kennzahl auf **Währung** fest.

Ergebnistabelle

product_type	**=sum(if(inmonthstodate($(vPeriod),sales_date,makedate(2022,12,24),0),sales_price,0))**
Produkt A	$186.48
Produkt B	$190.52
Produkt C	$220.43
Produkt D	$261.46

Die Funktion `inmonthstodate()` verwendet die Benutzereingabe als Argument zum Definieren der Größe des Startsegments des Jahres.

Die Funktion übergibt das Verkaufsdatum der einzelnen Produkte als das zweite Argument der Funktion `inmonthstodate()`. Da der 24. Dezember als drittes Argument in der Funktion `inmonthstodate()` verwendet wird, geben Produkte mit einem Verkaufsdatum im definierten Zeitraum bis einschließlich 24. Dezember einen booleschen Wert von TRUE zurück. Die Summenfunktion addiert die Verkäufe dieser Produkte.

inmonthtodate

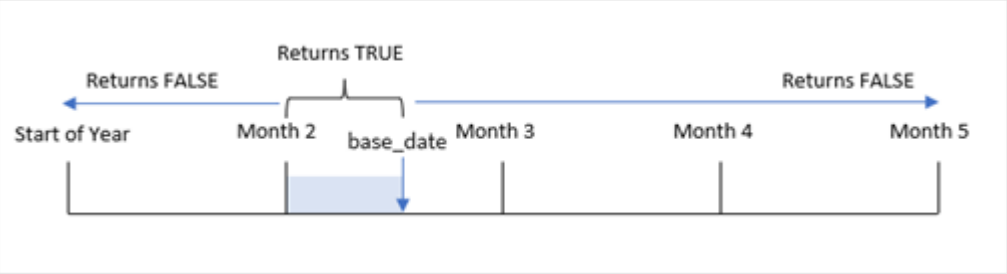
Liefert True, wenn **date** innerhalb des Teils des Monats liegt, der **basedate** enthält, und zwar bis einschließlich der letzten Millisekunde von **basedate**.

Syntax:

```
InMonthToDate (timestamp, base_date, period_no)
```

Rückgabe Datentyp: Boolesch

Diagramm der Funktion `inmonthtodate`.



Die Funktion `inmonthtodate()` identifiziert einen ausgewählten Monat als Segment. Die Startgrenze ist der Anfang des Monats. Die Endgrenze kann als ein späteres Datum im Monat festgelegt werden. Dann wird bestimmt, ob ein Satz Datumswerte in dieses Segment fällt oder nicht, und es wird ein boolescher Wert von TRUE oder FALSE zurückgegeben.

Argumente

Argument	Beschreibung
timestamp	Das Datum, das mit **base_date** verglichen werden soll.
base_date	Datum, das für die Interpretation des Monats verwendet wird.
period_no	Mit **period_no** kann ein anderer Beginn für den Monat festgelegt werden. **period_no** ist eine ganze Zahl, wobei 0 für den Monat steht, der **base_date** enthält. Negative Werte von **period_no** stehen für vorangehende, positive Werte für nachfolgende Monate.

Verwendung

Die Funktion `inmonthtodate()` gibt einen booleschen Wert zurück. In der Regel wird dieser Funktionstyp als Bedingung in einer `if expression` verwendet. Die Funktion `inmonthtodate()` gibt eine Aggregierung oder Berechnung zurück, abhängig davon, ob ein Datum in den Monat bis einschließlich zum betreffenden Datum fällt.

Beispielsweise kann die Funktion `inmonthtodate()` verwendet werden, um alle in einem Monat bis zu einem bestimmten Datum gefertigten Geräte zu identifizieren.

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Funktionsbeispiele

Beispiel	Ergebnis
`inmonthtodate ('01/25/2013', '25/01/2013', 0)`	Gibt True zurück
`inmonthtodate ('01/25/2013', '24/01/2013', 0)`	Gibt False zurück
`inmonthtodate ('01/25/2013', '28/02/2013', -1)`	Gibt True zurück

Beispiel 1 – keine zusätzlichen Argumente

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz, der eine Reihe von Transaktionen für 2022 enthält, wird in eine Tabelle namens „Transactions“ geladen.
- Ein Datumsfeld wird im Format DateFormat der Systemvariablen MM/DD/YYYY bereitgestellt.
- Ein vorangehender load-Befehl, der Folgendes enthält:
    - Die Funktion inmonthtodate(), die als Feld „in_month_to_date“ festgelegt ist. Dies bestimmt, welche Transaktionen zwischen dem 1. und 26. Juli 2022 stattgefunden haben.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
    *,
    inmonthtodate(date,'07/26/2022', 0) as in_month_to_date
    ;
Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `in_month_to_date`

Ergebnistabelle

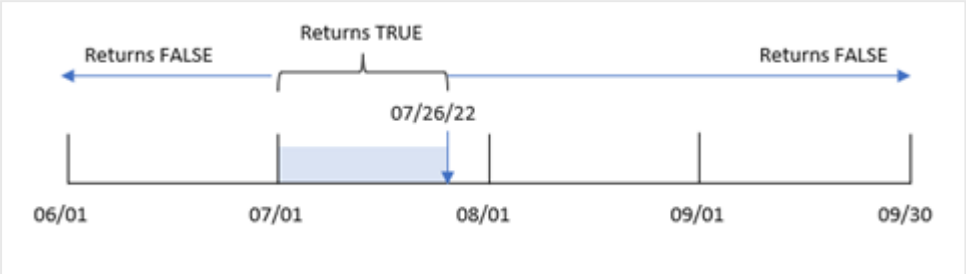
date	in_month_to_date
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	0
4/1/2022	0
5/7/2022	0
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	-1
7/22/2022	-1
7/23/2022	-1
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Das Feld „`in_month_to_date`“ wird in der vorangehenden load-Anweisung mithilfe der Funktion `inmonthtodate()` erstellt.

Das erste Argument identifiziert, welches Feld ausgewertet wird. Das zweite Argument ist ein hartcodiertes Datum, der 26. Juli. Dies ist das `base_date`. Dieses Argument `base_date` identifiziert, welcher Monat segmentiert wird, und bestimmt die Endgrenze dieses Segments.

Eine `period_no` von `0` ist das abschließende Argument, was bedeutet, dass die Funktion keine Monate vor oder nach dem segmentierten Monat vergleicht.

Diagramm der Funktion `inmonthtodate` ohne zusätzlichen Argumente.



Daher gibt jede Transaktion, die zwischen dem 1. Januar und 26. Juli stattfindet, ein boolesches Ergebnis von TRUE zurück. Jede Transaktion, die im Juli nach dem 26. Juli stattfindet, gibt ein boolesches Ergebnis von FALSE zurück, ebenso wie jede Transaktion in den anderen Monaten des Jahres.

Beispiel 2 – period_no

Ladeskript und Ergebnisse

Übersicht

Es werden derselbe Datensatz und dasselbe Szenario wie im ersten Beispiel verwendet.

In diesem Beispiel besteht die Aufgabe darin, ein Feld „`six_months_prior`“ zu erstellen, das bestimmt, welche Transaktionen volle sechs Monate vor dem 1. Juli bis 26. Juli stattfanden.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
    *,
    inmonthtodate(date,'07/26/2022', -6) as six_months_prior
    ;
Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
```

```
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `six_months_prior`

Ergebnistabelle

date	six_months_prior
1/7/2022	-1
1/19/2022	-1
2/5/2022	0
2/28/2022	0
3/16/2022	0
4/1/2022	0
5/7/2022	0
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0

date	six_months_prior
9/26/2022	0
10/14/2022	0
10/29/2022	0

Da -6 als das Argument period_no in der Funktion inmonthtodate() verwendet wird, verschieben sich die Grenzen des Vergleichsmonats um sechs Monate. Anfänglich liegt das Monatssegment zwischen dem 1. und dem 26. Juli. Die period_no versetzt dann dieses Segment um negative sechs Monate, und die Datumsgrenzen werden zum 1. bis 26. Januar verschoben.

Diagramm der Funktion inmonthtodate, wobei „period_no“ auf -6 festgelegt ist.

Hier gibt jede Transaktion, die zwischen dem 1. Januar und 26. Januar stattfindet, ein boolesches Ergebnis von TRUE zurück.

Beispiel 3 – Diagrammbeispiel

Ladeskript und Diagrammformel

Überblick

Es werden derselbe Datensatz und dasselbe Szenario wie im ersten Beispiel verwendet.

In diesem Beispiel wird der unveränderte Datensatz in die App geladen. Die Aufgabe besteht im Erstellen einer Berechnung, de bestimmt, ob Transaktionen zwischen dem 1. Juli und dem 26. Juli stattfanden. Sie wird als Kennzahl in einem Diagrammobjekt der App erstellt.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
```

```
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu:

```
date
```

Um zu berechnen, ob Transaktionen zwischen dem 1. und 26. Juli stattfanden, erstellen Sie die folgende Kennzahl:

```
=inmonthtodate(date,'07/26/2022', 0)
```

Ergebnistabelle

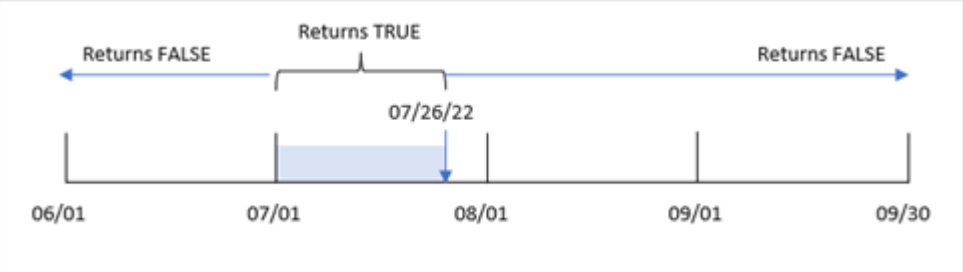
date	=inmonthtodate(date,'07/26/2022', 0)
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	0
4/1/2022	0
5/7/2022	0
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	-1

date	=inmonthtodate(date,'07/26/2022', 0)
7/22/2022	-1
7/23/2022	-1
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Das Feld „in_month_to_date“ wird als Kennzahl im Diagramm mithilfe der Funktion `inmonthtodate()` erstellt.

Das erste Argument identifiziert, welches Feld ausgewertet wird. Das zweite Argument ist ein hartcodiertes Datum, der 26. Juli. Dies ist das `base_date`. Dieses Argument „base_date“ identifiziert, welcher Monat segmentiert wird und legt die Endgrenze des Segments fest. Eine `period_no` von 0 ist das abschließende Argument. Das bedeutet, dass die Funktion keine Monate vor oder nach dem segmentierten Monat vergleicht.

Diagramm der Funktion `inmonthtodate` ohne zusätzlichen Argumente.



Daher gibt jede Transaktion, die zwischen dem 1. Januar und 26. Juli stattfindet, ein boolesches Ergebnis von TRUE zurück. Jede Transaktion, die im Juli nach dem 26. Juli stattfindet, gibt ein boolesches Ergebnis von FALSE zurück, ebenso wie jede Transaktion in den anderen Monaten des Jahres.

Beispiel 4 – Szenario

Ladeskript und Ergebnisse

Übersicht

In diesem Beispiel wird ein Datensatz in eine Tabelle namens „Products“ geladen. Die Tabelle enthält die folgenden Felder:

- Produkt-ID
- Herstellungsdatum

- Selbstkosten

Aufgrund eines Maschinenfehlers waren die im Monat Juli 2022 gefertigten Produkte mangelhaft. Das Problem wurde am 27. Juli 2022 behoben.

Der Endbenutzer möchte ein Diagramm haben, das nach Monat den Status der hergestellten Produkte angibt und zeigt, welche „mangelhaft“ (boolesch TRUE) und welche „einwandfrei“ (boolesch FALSE) waren und was die in diesem Monat gefertigten Produkte gekostet haben.

Ladeskript

```
Products:
Load
*
Inline
[
product_id,manufacture_date,cost_price
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `=monthname(manufacture_date)`
- `=if(Inmonthtodate(manufacture_date,makedate(2022,07,26),0),'Defective','Faultless')`

Um die Summe der Produktkosten zu berechnen, erstellen Sie die folgende Kennzahl:

```
=sum(cost_price)
```

Legen Sie das **Zahlenformat** der Kennzahl auf **Währung** fest.

Ergebnistabelle

monthname (manufacture_date)	if(Inmonthtodate(manufacture_date,makedate (2022,07,26),0),'Defective','Faultless')	Sum(cost_ price)
Jan 2022	Faultless	$54.40
Feb 2022	Faultless	$145.69
Mar 2022	Faultless	$53.80
Apr 2022	Faultless	$82.06
May 2022	Faultless	$127.60
Jun 2022	Faultless	$141.82
Jul 2022	Defective	$144.66
Jul 2022	Faultless	$69.98
Aug 2022	Faultless	$147.46
Sep 2022	Faultless	$84.21
Oct 2022	Faultless	$163.91

Die Funktion `inmonthtodate()` gibt einen booleschen Wert zurück, wenn sie das Herstellungsdatum der einzelnen Produkte auswertet.

Für die Datumswerte, die einen booleschen Wert von TRUE zurückgeben, wird das Produkt als „mangelhaft“ markiert. Alle Produkte, die einen Wert von FALSE zurückgeben und daher nicht im Monat bis einschließlich 26. Juli gefertigt wurden, werden als „einwandfrei“ markiert.

inquarter

Diese Funktion liefert True, wenn **timestamp** innerhalb des Quartals liegt, das **base_date** enthält.

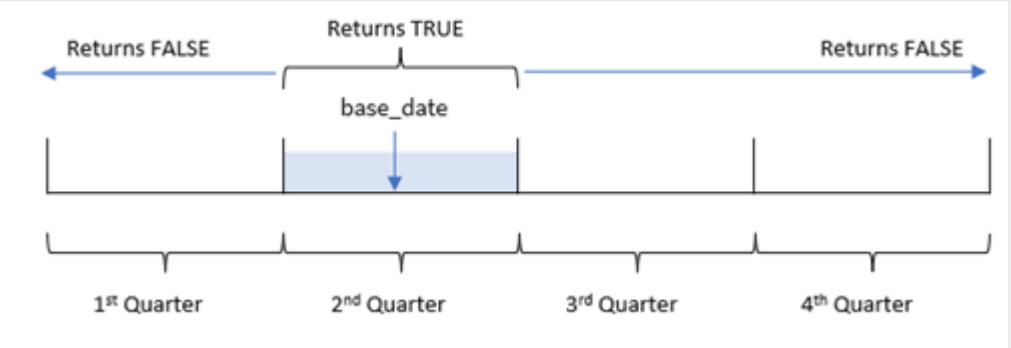
Syntax:

```
InQuarter (timestamp, base_date, period_no[, first_month_of_year])
```

Rückgabe Datentyp: Boolesch

In Qlik Sense wird der boolesche Wert „wahr“ durch -1 dargestellt, der Wert „falsch“ durch 0.

Diagramm des Bereichs der Funktion `inquarter()`



In anderen Worten teilt die Funktion `inquarter()` das Jahr in vier gleiche Quartale zwischen dem 1. Januar und dem 31. Dezember auf. Sie können das Argument `first_month_of_year` verwenden, um zu ändern, welcher Monat in Ihrer App als der erste gilt. Die Quartale werden entsprechend diesem Argument geändert. Mit `base_date` identifiziert die Funktion, welches Quartal als Vergleichselement für die Funktion verwendet werden soll. Abschließend gibt die Funktion ein boolesches Ergebnis zurück, wenn die Datumswerte mit diesem Quartalssegment verglichen werden.

Verwendung

Die Funktion `inquarter()` gibt einen booleschen Wert zurück. In der Regel wird dieser Funktionstyp als Bedingung in einer `if expression` verwendet. Dies gibt eine Aggregierung oder Berechnung zurück, abhängig davon, ob ein Datum in das ausgewählte Quartal fällt.

Beispielsweise kann die Funktion `inquarter()` verwendet werden, um alle Geräte zu identifizieren, die in einem Quartalssegment hergestellt wurden, gestützt auf das Datum, an dem das jeweilige Gerät hergestellt wurde.

Argumente

Argument	Beschreibung
timestamp	Das Datum, das mit **base_date** verglichen werden soll.
base_date	Datum, das für die Interpretation des Quartals verwendet wird.
period_no	Mit **period_no** kann der Beginn für das Quartal festgelegt werden. **period_no** ist eine ganze Zahl, wobei 0 für das Quartal steht, das **base_date** enthält. Negative Werte von **period_no** stehen für vorangehende, positive Werte für nachfolgende Quartale.
first_month_of_year	Wenn Sie mit abweichenden Geschäftsjahren arbeiten möchten, definieren Sie mit einer Zahl zwischen 2 und 12 für **first_month_of_year** einen anderen Startmonat.

Sie können die folgenden Werte verwenden, um den ersten Monat des Jahres im Argument `first_month_of_year` festzulegen.

Werte für first_month_of_year

Monat	Wert
Februar	2
März	3
April	4
Mai	5
Juni	6
Juli	7
August	8
September	9
Oktober	10
November	11
Dezember	12

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Funktionsbeispiele

Beispiel	Ergebnis
inquarter ('01/25/2013', '01/01/2013', 0)	Gibt TRUE zurück
inquarter ('01/25/2013', '04/01/2013', 0)	Gibt FALSE zurück
inquarter ('01/25/2013', '01/01/2013', -1)	Gibt FALSE zurück
inquarter ('12/25/2012', '01/01/2013', -1)	Gibt TRUE zurück
inquarter ('01/25/2013', '03/01/2013', 0, 3)	Gibt FALSE zurück
inquarter ('03/25/2013', '03/01/2013', 0, 3)	Gibt TRUE zurück

Beispiel 1 – keine zusätzlichen Argumente

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz enthält eine Reihe von Transaktionen für 2022 und wird in eine Tabelle namens „`Transactions`“ geladen.
- Ein vorangehender load-Befehl enthält die Funktion `inquarter()`, die als das Feld „`in_quarter`“ festgelegt wird und bestimmt, welche Transaktionen im gleichen Quartal wie der 15. Mai 2022 stattfanden.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
       *,
       inquarter (date,'05/15/2022', 0) as in_quarter
       ;
Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

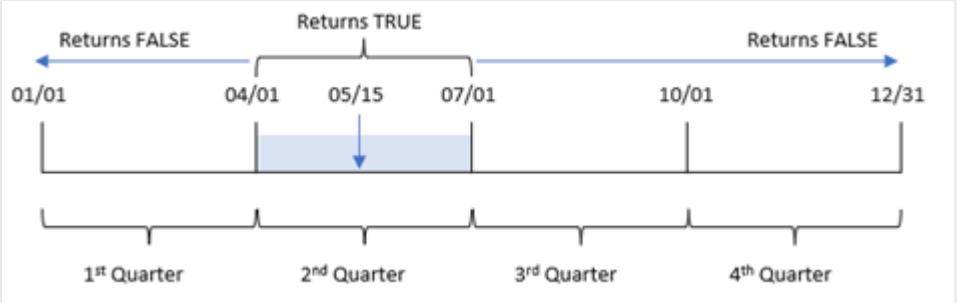
- `date`
- `in_quarter`

Ergebnistabelle

date	in_quarter
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	0
4/1/2022	-1
5/7/2022	-1
5/16/2022	-1
6/15/2022	-1
6/26/2022	-1
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Das Feld „`in_quarter`“ wird in der vorangehenden load-Anweisung mithilfe der Funktion `inquarter()` erstellt. Das erste Argument identifiziert, welches Feld ausgewertet wird. Das zweite Argument ist ein hartcodiertes Datum, der 15. Mai, das identifiziert, welches Quartal als Vergleichselement definiert wird. Eine `period_no` von `0` ist das abschließende Argument, das dafür sorgt, dass die Funktion `inquarter()` keine Quartale vor oder nach dem segmentierten Quartal vergleicht.

Diagramm der Funktion `inquarter()` mit dem 15. Mai als Basisdatum



Jede Transaktion, die zwischen dem 1. April und dem Ende des 30. Junis stattfindet, gibt ein boolesches Ergebnis von TRUE zurück.

Beispiel 2 – period_no

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz enthält eine Reihe von Transaktionen für 2022 und wird in eine Tabelle namens „`Transactions`“ geladen.
- Eine vorangehende load-Anweisung enthält die Funktion `inquarter()`, die als das Feld „`previous_quarter`“ festgelegt wird und bestimmt, welche Transaktionen im Quartal vor dem Quartal des 15. Mai 2022 stattfanden.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
       *,
       inquarter (date,'05/15/2022', -1) as previous_qtr
       ;
Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
```

```
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `previous_qtr`

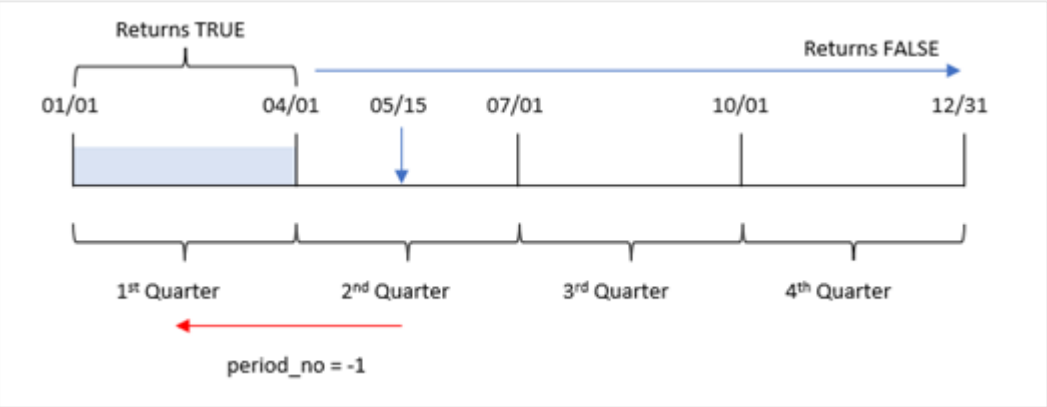
Ergebnistabelle

date	previous_qtr
1/7/2022	-1
1/19/2022	-1
2/5/2022	-1
2/28/2022	-1
3/16/2022	-1
4/1/2022	0
5/7/2022	0
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0

date	previous_qtr
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Da -1 als Argument `period_no` in der Funktion `inquarter()` verwendet wird, werden die Grenzen des Vergleichsquartals um ein ganzes Quartal zurück verschoben. Der 15. Mai fällt in das zweite Quartal des Jahres, sodass das Segment anfänglich dem Zeitraum vom 1. April bis zum 30. Juni entspricht. Die `period_no` verschiebt dann dieses Segment um negative drei Monate, sodass die Datumsgrenzen der 1. Januar bis zum 30. März werden.

Diagramm der Funktion `inquarter()` mit dem 15. Mai als Basisdatum



Daher gibt jede Transaktion, die zwischen dem 1. Januar und dem 30. März stattfindet, ein boolesches Ergebnis von TRUE zurück.

Beispiel 3 – first_month_of_year

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz enthält eine Reihe von Transaktionen für 2022 und wird in eine Tabelle namens „`Transactions`“ geladen.
- Eine vorangehende load-Anweisung enthält die Funktion `inquarter()`, die als das Feld „`in_quarter`“ festgelegt wird und bestimmt, welche Transaktionen im gleichen Quartal wie der 15. Mai 2022 stattfanden.

In diesem Beispiel legt aber die Organisationsrichtlinie den März als ersten Monat des Geschäftsjahres fest.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
       *,
       inquarter (date,'05/15/2022', 0, 3) as in_quarter
       ;
Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `previous_qtr`

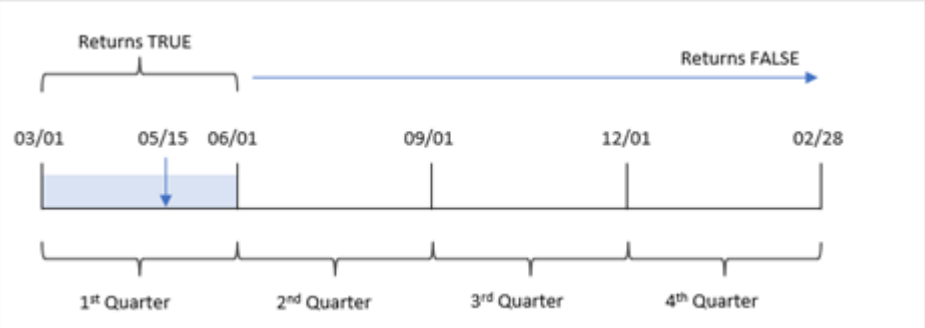
Ergebnistabelle

date	previous_qtr
1/7/2022	0
1/19/2022	0

date	previous_qtr
2/5/2022	0
2/28/2022	0
3/16/2022	-1
4/1/2022	-1
5/7/2022	-1
5/16/2022	-1
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Da `3` als Argument `first_month_of_year` in der Funktion `inquarter()` verwendet wird, wird der 1. März als Jahresbeginn festgelegt und dann das Jahr in Quartale unterteilt. Daher lauten die Quartalssegmente Mär-Mai, Jun-Aug, Sep-Nov und Dez-Feb. Das `base_date` vom 15. Mai legt das Quartal Mär-Mai als Vergleichsquartal für die Funktion fest.

Diagramm der Funktion `inquarter()` mit März als erstem Monat des Jahres



Daher gibt jede Transaktion, die zwischen dem 1. März und dem 31. Mai stattfindet, ein boolesches Ergebnis von TRUE zurück.

Beispiel 4 – Diagrammobjektbeispiel

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz enthält eine Reihe von Transaktionen für 2022 und wird in eine Tabelle namens „`Transactions`“ geladen.
- Eine vorangehende load-Anweisung enthält die Funktion `inquarter()`, die als das Feld „`in_quarter`“ festgelegt wird und bestimmt, welche Transaktionen im gleichen Quartal wie der 15. Mai 2022 stattfanden.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu:

- `date`

Erstellen Sie die folgende Kennzahl, um zu berechnen, ob Transaktionen im selben Quartal wie der 15. Mai stattfanden:

```
=inquarter(date,'05/15/2022', 0)
```

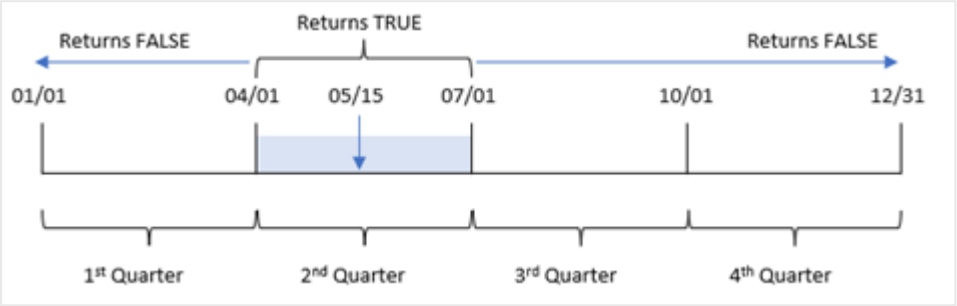
Ergebnistabelle

date	in_quarter
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	0
4/1/2022	-1
5/7/2022	-1
5/16/2022	-1
6/15/2022	-1
6/26/2022	-1
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Die Kennzahl „`in_quarter`“ wird im Diagramm anhand der Funktion `inquarter()` erstellt. Das erste Argument identifiziert, welches Feld ausgewertet wird. Das zweite Argument ist ein hartcodiertes Datum, der 15. Mai, das

identifiziert, welches Quartal als Vergleichselement definiert wird. Eine `period_no` von `0` ist das abschließende Argument, das dafür sorgt, dass die Funktion `inquarter()` keine Quartale vor oder nach dem segmentierten Quartal vergleicht.

Diagramm der Funktion `inquarter()` mit dem 15. Mai als Basisdatum



Jede Transaktion, die zwischen dem 1. April und dem Ende des 30. Junis stattfindet, gibt ein boolesches Ergebnis von TRUE zurück.

Beispiel 5 – Szenario

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Datensatz, der in eine Tabelle namens „`Products`“ geladen wird
- Die Tabelle enthält die folgenden Felder:
    - product ID
    - product type
    - manufacture date
    - cost price

Es wurde festgestellt, dass aufgrund eines Ausrüstungsfehlers die in der Woche vom 15. Mai 2022 hergestellten Produkte mangelhaft waren. Der Endbenutzer möchte ein Diagramm, das nach Namen des Quartals den Status der hergestellten Produkte angibt und zeigt, welche „mangelhaft“ und welche „einwandfrei“ waren und was die in diesem Quartal gefertigten Produkte gekostet haben.

Ladeskript

```
Products:
Load
*
Inline
[
product_id,manufacture_date,cost_price
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
```

```
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu:

```
=quartername(manufacture_date)
```

Erstellen Sie die folgenden Kennzahlen:

- `=if(only(InQuarter(manufacture_date,makedate(2022,05,15),0)),'Defective','Faultless')`, um zu identifizieren, welche Produkte mangelhaft und welche einwandfrei sind. Verwenden Sie dafür die Funktion `inquarter()`.
- `=sum(cost_price)`, um die Summe der Kosten für jedes Produkt zu zeigen.

Gehen Sie folgendermaßen vor:

1. Legen Sie das **Zahlenformat** der Kennzahl auf **Währung** fest.
2. Deaktivieren Sie unter **Darstellung** die Option **Gesamtwerte**.

Ergebnistabelle

quartername (manufacture_date)	**=if(only(InQuarter(manufacture_date,makedate (2022,05,15),0)),'Defective','Faultless')**	**Sum(cost_ price)**
Jan-Mar 2022	Faultless	253.89
Apr-Jun 2022	Defective	351.48
Jul-Sep 2022	Faultless	446.31
Oct-Dec 2022	Faultless	163.91

Die Funktion `inquarter()` gibt einen booleschen Wert zurück, wenn sie das Herstellungsdatum der einzelnen Produkte auswertet. Für alle im Quartal, in dem der 15. Mai liegt, hergestellten Produkte gibt die Funktion `inquarter()` einen booleschen Wert von TRUE zurück und markiert die Produkte als „mangelhaft“. Alle Produkte, die einen Wert von FALSE zurückgeben und daher nicht in diesem Quartal hergestellt wurden, werden als „einwandfrei“ markiert.

inquartertodate

Diese Funktion liefert True, wenn **timestamp** in dem Teil des Quartals liegt, der **base_date** enthält, und zwar bis einschließlich der letzten Millisekunde von **base_date**.

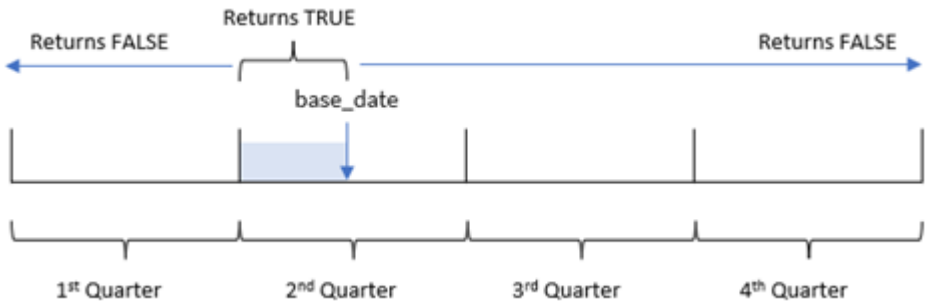
Syntax:

```
InQuarterToDate (timestamp, base_date, period_no [, first_month_of_year])
```

Rückgabe Datentyp: Boolesch

> *In Qlik Sense wird der boolesche Wert „wahr“ durch -1 dargestellt, der Wert „falsch“ durch 0.*

Diagramm der Funktion „inquartertodate“



Die Funktion `inquartertodate()` unterteilt das Jahr in vier gleiche Quartale zwischen dem 1. Januar und dem 31. Dezember (bzw. dem benutzerdefinierten Jahresbeginn und dem entsprechenden Enddatum). Unter Verwendung des `base_date` unterteilt die Funktion dann ein bestimmtes Quartal, wobei das `base_date` sowohl das Quartal als auch das maximal zulässige Datum für dieses Quartalssegment definiert. Abschließend gibt die Funktion ein boolesches Ergebnis zurück, wenn die vordefinierten Datumswerte mit diesem Segment verglichen werden.

Argumente

Argument	Beschreibung
timestamp	Das Datum, das mit **base_date** verglichen werden soll.
base_date	Datum, das für die Interpretation des Quartals verwendet wird.
period_no	Mit **period_no** kann der Beginn für das Quartal festgelegt werden. **period_no** ist eine ganze Zahl, wobei 0 für das Quartal steht, das **base_date** enthält. Negative Werte von **period_no** stehen für vorangehende, positive Werte für nachfolgende Quartale.

Argument	Beschreibung
first_month_of_year	Wenn Sie mit abweichenden Geschäftsjahren arbeiten möchten, definieren Sie mit einer Zahl zwischen 2 und 12 für **first_month_of_year** einen anderen Startmonat.

Verwendung

Die Funktion `inquartertodate()` gibt einen booleschen Wert zurück. In der Regel wird dieser Funktionstyp als Bedingung in einem `if`-Ausdruck verwendet. Die Funktion `inquartertodate()` gibt eine Aggregierung oder Berechnung zurück, abhängig davon, ob ein ausgewertetes Datum in das Quartal bis einschließlich zum betreffenden Datum fällt.

Beispielsweise kann die Funktion `inquartertodate()` verwendet werden, um alle in einem Quartal bis zu einem bestimmten Datum gefertigten Geräte zu identifizieren.

Funktionsbeispiele

Beispiel	Ergebnis
`inquartertodate ('01/25/2013', '03/25/2013', 0)`	Gibt `TRUE` zurück, da der Wert von `timestamp`, 01/25/2013, innerhalb des Dreimonatszeitraums vom 01/01/2013 bis zum 03/25/2013 liegt, in den der Wert von `base_date`, 03/25/2013, fällt.
`inquartertodate ('04/26/2013', '03/25/2013', 0)`	Gibt `FALSE` zurück, da 04/26/2013 außerhalb des Zeitraums aus dem vorigen Beispiel liegt.
`inquartertodate ('02/25/2013', '06/09/2013', -1)`	Gibt `TRUE` zurück, da der Wert von `period_no`, -1, den Suchzeitraum um einen Dreimonatszeitraum zurück verschiebt (ein Quartal des Jahres). Dadurch ist der Suchzeitraum 01/01/2013 bis 03/09/2013.
`inquartertodate ('03/25/2006', '04/15/2006', 0, 2)`	Gibt `TRUE` zurück, da der Wert von `first_month_of_year` auf 2 festgelegt ist, wodurch der Suchzeitraum 02/01/2006 bis 04/15/2006 anstelle von 04/01/2006 bis 04/15/2006 ist.

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Beispiel 1 – Keine zusätzlichen Argumente

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz enthält eine Reihe von Transaktionen für 2022 und wird in eine Tabelle namens `Transactions` geladen.
- Das Datumsfeld wird im Format der Systemvariablen `DateFormat` (MM/TT/JJJJ) bereitgestellt.
- Die Erstellung eines Felds `in_quarter_to_date`, das bestimmt, welche Transaktionen im Quartal bis zum 15. Mai 2022 stattgefunden haben.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        inquartertodate(date,'05/15/2022', 0) as in_quarter_to_date
        ;
Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

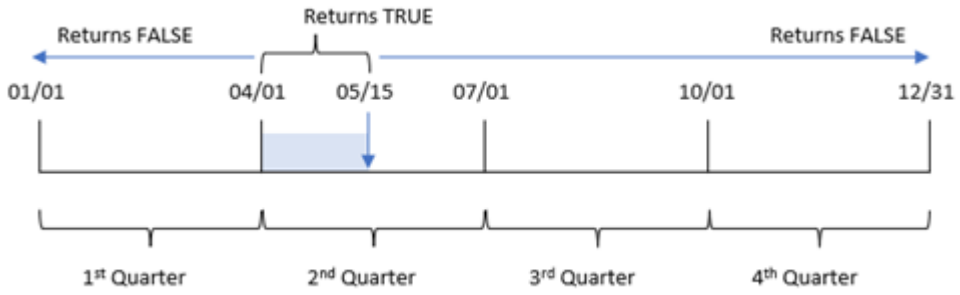
- `date`
- `in_quarter_to_date`

Ergebnistabelle

date	in_quarter_to_date
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	0
4/1/2022	-1
5/7/2022	-1
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Das Feld „`in_quarter_to_date`“ wird im vorangehenden load-Befehl mithilfe der Funktion `inquartertodate()` erstellt. Das erste angegebene Argument identifiziert, welches Feld ausgewertet wird. Das zweite Argument ist ein hardcodiertes Datum für den 15. Mai. Dies ist das `base_date`, das identifiziert, welches Quartal segmentiert wird, und die Endbegrenzung dieses Segments definiert. Eine `period_no` von 0 ist das abschließende Argument, was bedeutet, dass die Funktion keine Quartale vor oder nach dem segmentierten Quartal vergleicht.

Diagramm der Funktion „inquartertodate“, keine zusätzlichen Argumente



Jede Transaktion, die zwischen dem 1. April und dem 15. Mai stattfindet, gibt ein boolesches Ergebnis von TRUE zurück. Transaktionen mit einem Datum am oder nach dem 16. Mai geben FALSE zurück, ebenso wie alle Transaktionen vor dem 1. April.

Beispiel 2 – period_no

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Derselbe Datensatz und dasselbe Szenario wie im ersten Beispiel.
- Die Erstellung eines Felds `previous_qtr_to_date`, das bestimmt, welche Transaktionen ein ganzes Quartal vor dem Quartalssegment stattfanden, das am 15. Mai 2022 endet.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        inquartertodate(date,'05/15/2022', -1) as previous_qtr_to_date
        ;
Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
```

```
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `previous_qtr_to_date`

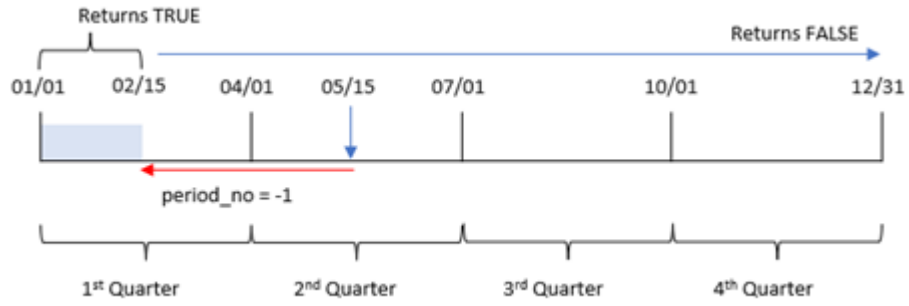
Ergebnistabelle

date	previous_qtr_to_date
1/7/2022	-1
1/19/2022	-1
2/5/2022	-1
2/28/2022	0
3/16/2022	0
4/1/2022	0
5/7/2022	0
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0

date	previous_qtr_to_date
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Ein `period_no`-Wert von -1 gibt an, dass die Funktion `inquartertodate ()` das Eingabequartalssegment mit dem vorherigen Quartal vergleicht. Der 15. Mai fällt in das zweite Quartal des Jahres, sodass das Segment anfänglich dem Zeitraum vom 1. April bis zum 15. Mai entspricht. Die `period_no` verschiebt dann dieses Segment um drei Monate zurück, sodass die Datumsgrenzen der 1. Januar bis zum 15. Februar werden.

Diagramm der Funktion „inquartertodate“, Beispiel „period_no“



Daher gibt jede Transaktion, die zwischen dem 1. Januar und 15. Februar stattfindet, ein boolesches Ergebnis von `TRUE` zurück.

Beispiel 3 – first_month_of_year

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Derselbe Datensatz und dasselbe Szenario wie im ersten Beispiel.
- Die Erstellung eines Felds `in_quarter_to_date`, das bestimmt, welche Transaktionen im gleichen Quartal bis zum 15. Mai 2022 stattgefunden haben.

In diesem Beispiel wird März als erster Monat des Geschäftsjahrs festgelegt.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        inquartertodate(date,'05/15/2022', 0,3) as in_quarter_to_date
        ;
Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `in_quarter_to_date`

Ergebnistabelle

date	in_quarter_to_date
1/7/2022	0
1/19/2022	0
2/5/2022	0

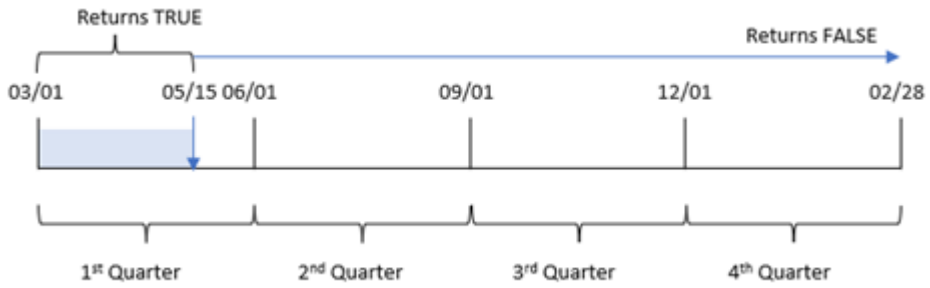
date	in_quarter_to_date
2/28/2022	0
3/16/2022	-1
4/1/2022	-1
5/7/2022	-1
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Indem 3 als Argument `first_month_of_year` in der Funktion `inquartertodate()` verwendet wird, beginnt die Funktion das Jahr am 1. März und unterteilt dann das Jahr in Quartale. Daher lauten die Quartalssegmente wie folgt:

- März bis Mai
- Juni bis August
- September bis November
- Dezember bis Februar

Das `base_date` vom 15. Mai segmentiert dann das Quartal März bis Mai, indem seine Endgrenze als 15. Mai festgelegt wird.

Diagramm der Funktion „inquartertodate“, Beispiel „first_month_of_year“



Daher gibt jede Transaktion, die zwischen dem 1. März und dem 15. Mai stattfindet, ein boolesches Ergebnis von TRUE zurück, während Transaktionen mit Datum außerhalb dieser Grenzen einen Wert von FALSE zurückgeben.

Beispiel 4 – Diagrammobjektbeispiel

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript verwendet den gleichen Datensatz und das gleiche Szenario wie das erste Beispiel. In diesem Beispiel wird jedoch der unveränderte Datensatz in die Anwendung geladen. Die Berechnung, die bestimmt, ob Transaktionen im gleichen Quartal wie der 15. Mai stattfanden, wird als Kennzahl in einem Diagrammobjekt erstellt.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
```

```
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu:`date`.

Erstellen Sie die folgende Kennzahl:

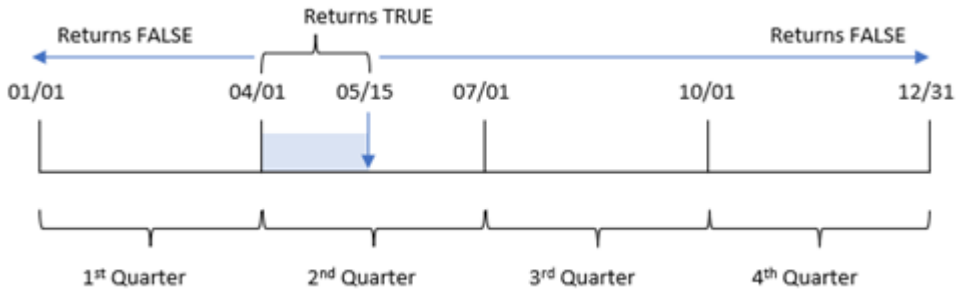
```
=inquartertodate(date,'05/15/2022', 0)
```

Ergebnistabelle

date	**=inquartertodate(date,'05/15/2022', 0)**
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	0
4/1/2022	-1
5/7/2022	-1
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Die Kennzahl `in_quarter_to_date` wird in einem Diagrammobjekt anhand der Funktion `inquartertodate()` erstellt. Das erste Argument ist das Datumsfeld, das ausgewertet wird. Das zweite Argument ist ein hardcodiertes Datum für den 15. Mai. Dies ist das `base_date`, das identifiziert, welches Quartal segmentiert wird, und definiert die Endbegrenzung dieses Segments. Eine `period_no` von 0 ist das abschließende Argument, was bedeutet, dass die Funktion keine Quartale vor oder nach dem segmentierten Quartal vergleicht.

Diagramm der Funktion „inquartertodate“, Diagrammobjektbeispiel



Jede Transaktion, die zwischen dem 1. April und dem 15. Mai stattfindet, gibt ein boolesches Ergebnis von TRUE zurück. Transaktionen mit einem Datum am oder nach dem 16. Mai geben FALSE zurück, ebenso wie alle Transaktionen vor dem 1. April.

Beispiel 5 – Szenario

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz, der in eine Tabelle namens `Products` geladen wird.
- Informationen bezüglich Produkt-ID, Herstellungsdatum und Selbstkosten.

Am 15. Mai 2022 wurde ein Maschinenfehler im Herstellungsprozess identifiziert und behoben. Produkte, die in diesem Quartal bis zu diesem Datum hergestellt wurden, sind mangelhaft. Der Endbenutzer möchte ein Diagrammobjekt haben, das nach Quartalsnamen den Status der Produkte als „mangelhaft“ und als „einwandfrei“ und die Kosten der in diesem Quartal bisher gefertigten Produkte angibt.

Ladeskript

```
Products:
Load
*
Inline
[
product_id,manufacture_date,cost_price
8188,'1/19/2022',37.23
```

```
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Ergebnisse

Gehen Sie folgendermaßen vor:

1. Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle. Erstellen Sie eine Dimension, um die Quartalsnamen anzuzeigen:
   `=quartername(manufacture_date)`
2. Erstellen Sie dann eine Dimension, um zu identifizieren, welche Produkte mangelhaft und welche einwandfrei sind:
   `=if(inquartertodate(manufacture_date,makedate(2022,05,15),0),'Defective','Faultless')`
3. Erstellen Sie eine Kennzahl zum Summieren des `cost_price` der Produkte:
   `=sum(cost_price)`
4. Legen Sie das **Zahlenformat** der Kennzahl auf **Währung** fest.

Ergebnistabelle

quartername (manufacture_date)	**if(inquartertodate(manufacture_date,makedate (2022,05,15),0),'Defective','Faultless')**	**Sum(cost_ price)**
Jan-Mar 2022	Faultless	$253.89
Apr-Jun 2022	Faultless	$229.03
Apr-Jun 2022	Defective	$122.45
Jul-Sep 2022	Faultless	$446.31
Oct-Dec 2022	Faultless	$163.91

Die Funktion `inquartertodate()` gibt einen booleschen Wert zurück, wenn sie das Herstellungsdatum der einzelnen Produkte auswertet. Für diejenigen, die einen booleschen Wert von TRUE zurückgeben, markiert sie die Produkte als `'Defective'`. Für jedes Produkt, das einen Wert von FALSE zurückgibt und somit nicht im Quartal bis einschließlich 15. Mai hergestellt wurde, werden die Produkte als `'Faultless'` markiert.

inweek

Diese Funktion liefert True, wenn **timestamp** innerhalb der Woche liegt, die **base_date** enthält.

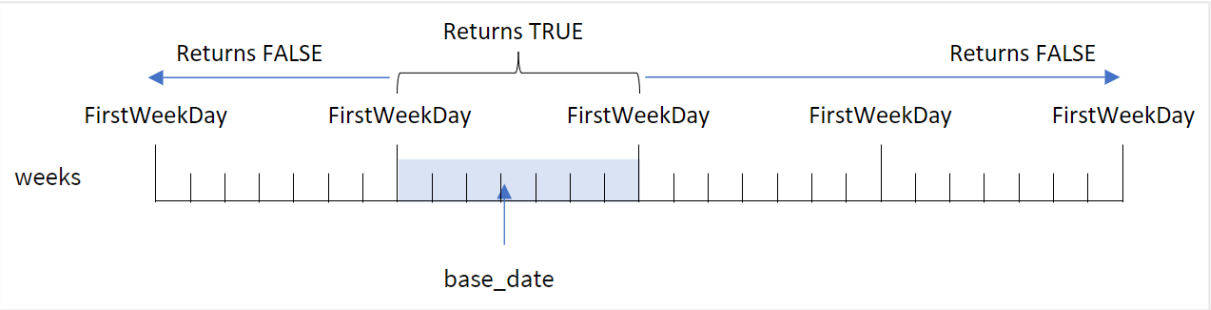
Syntax:

```
InWeek (timestamp, base_date, period_no[, first_week_day])
```

Rückgabe Datentyp: Boolesch

In Qlik Sense wird der boolesche Wert „wahr“ durch -1 dargestellt, der Wert „falsch“ durch 0.

Diagramm des Bereichs der Funktion `inweek()`



Die Funktion `inweek()` verwendet das Argument `base_date`, um zu identifizieren, in welchen Siebentagezeitraum das Datum fällt. Der Starttag der Woche basiert auf der Systemvariablen `FirstWeekDay`. Sie können aber auch den ersten Tag der Woche ändern, indem Sie das Argument `first_week_day` in der Funktion `inweek()` verwenden.

Nachdem die ausgewählte Woche definiert wurde, gibt die Funktion boolesche Ergebnisse zurück, wenn die vordefinierten Datumswerte mit diesem Wochensegment verglichen werden.

Verwendung

Die Funktion `inweek()` gibt einen booleschen Wert zurück. In der Regel wird dieser Funktionstyp als Bedingung in einer `if expression` verwendet. Die Funktion `inweek()` gibt eine Aggregierung oder Berechnung zurück, abhängig davon, ob ein ausgewertetes Datum in die Woche mit dem ausgewählten Datum des Arguments `base_date` fällt.

Beispielsweise kann die Funktion `inweek()` verwendet werden, um alle in einer bestimmten Woche gefertigten Geräte zu identifizieren.

Argumente

Argument	Beschreibung
timestamp	Das Datum, das mit **base_date** verglichen werden soll.
base_date	Datum, das für die Interpretation der Woche verwendet wird.

Argument	Beschreibung
period_no	Mit **period_no** kann der Beginn der Woche festgelegt werden. **period_no** ist eine ganze Zahl, wobei 0 für die Woche steht, die **base_date** enthält. Negative Werte von **period_no** stehen für vorangehende Wochen, positive Werte für nachfolgende Wochen.
first_week_day	Standardmäßig ist der erste Tag der Woche der Sonntag (wie durch die Systemvariable „FirstWeekDay“ definiert) und beginnt um Mitternacht zwischen Samstag und Sonntag. Der Parameter **first_week_day** überschreibt die Variable **FirstWeekDay**. Wenn der Wochenbeginn auf einen anderen Tag fallen soll, geben Sie ein Flag zwischen 0 und 6 an.

Werte für first_week_day

Tag	Wert
Montag	0
Dienstag	1
Mittwoch	2
Donnerstag	3
Freitag	4
Samstag	5
Sonntag	6

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Funktionsbeispiele

Beispiel	Ergebnis
`inweek ('01/12/2006', '01/14/2006', 0)`	Gibt TRUE zurück

Beispiel	Ergebnis
`inweek ('01/12/2006', '01/20/2006', 0 )`	Gibt FALSE zurück
`inweek ('01/12/2006', '01/14/2006', -1 )`	Gibt FALSE zurück
`inweek ('01/07/2006', '01/14/2006', -1)`	Gibt TRUE zurück
`inweek ('01/12/2006', '01/09/2006', 0, 3)`	Gibt FALSE zurück, weil `first_week_day` als 3 (Donnerstag) angegeben ist. Dadurch wird 01/12/2006 zum ersten Tag der Woche nach der Woche, die den 01/09/2006 enthält.

Die folgenden Themen können Sie bei der Arbeit mit dieser Funktion unterstützen:

Verwandte Themen

Thema	Standardkennzeichen/Wert	Beschreibung
FirstWeekDay (page 233)	6 / Sonntag	Definiert den Starttag jeder Woche.

Beispiel 1 – keine zusätzlichen Argumente

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Datensatz mit einer Reihe von Transaktionen für den Monat Januar 2022, der in eine Tabelle namens „`Transactions`“ geladen wird.
- Die Systemvariable `FirstWeekDay` ist auf 6 (Sonntag) festgelegt.
- Ein vorangehender load-Befehl, der Folgendes enthält:
  - Die Funktion `inweek()`, die als das Feld „`in_week`“ festgelegt ist, das bestimmt, welche Transaktionen in der Woche vom 14. Januar 2022 stattfanden.
  - Die Funktion `weekday()`, die als das Feld „`week_day`“ festgelegt ist, das zeigt, welcher Tag der Woche jedem Datum entspricht.

Ladeskript

```
SET FirstWeekDay=6;
SET DateFormat='MM/DD/YYYY';
```

```
Transactions:
    Load
       *,
       weekday(date) as week_day,
       inweek(date,'01/14/2022', 0) as in_week
       ;
Load
*
Inline
[
id,date,amount
8188,'01/02/2022',37.23
8189,'01/05/2022',17.17
8190,'01/06/2022',88.27
8191,'01/08/2022',57.42
8192,'01/09/2022',53.80
8193,'01/10/2022',82.06
8194,'01/11/2022',40.39
8195,'01/12/2022',87.21
8196,'01/13/2022',95.93
8197,'01/14/2022',45.89
8198,'01/15/2022',36.23
8199,'01/16/2022',25.66
8200,'01/17/2022',82.77
8201,'01/18/2022',69.98
8202,'01/26/2022',76.11
8203,'01/27/2022',25.12
8204,'01/28/2022',46.23
8205,'01/29/2022',84.21
8206,'01/30/2022',96.24
8207,'01/31/2022',67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `week_day`
- `in_week`

Ergebnistabelle

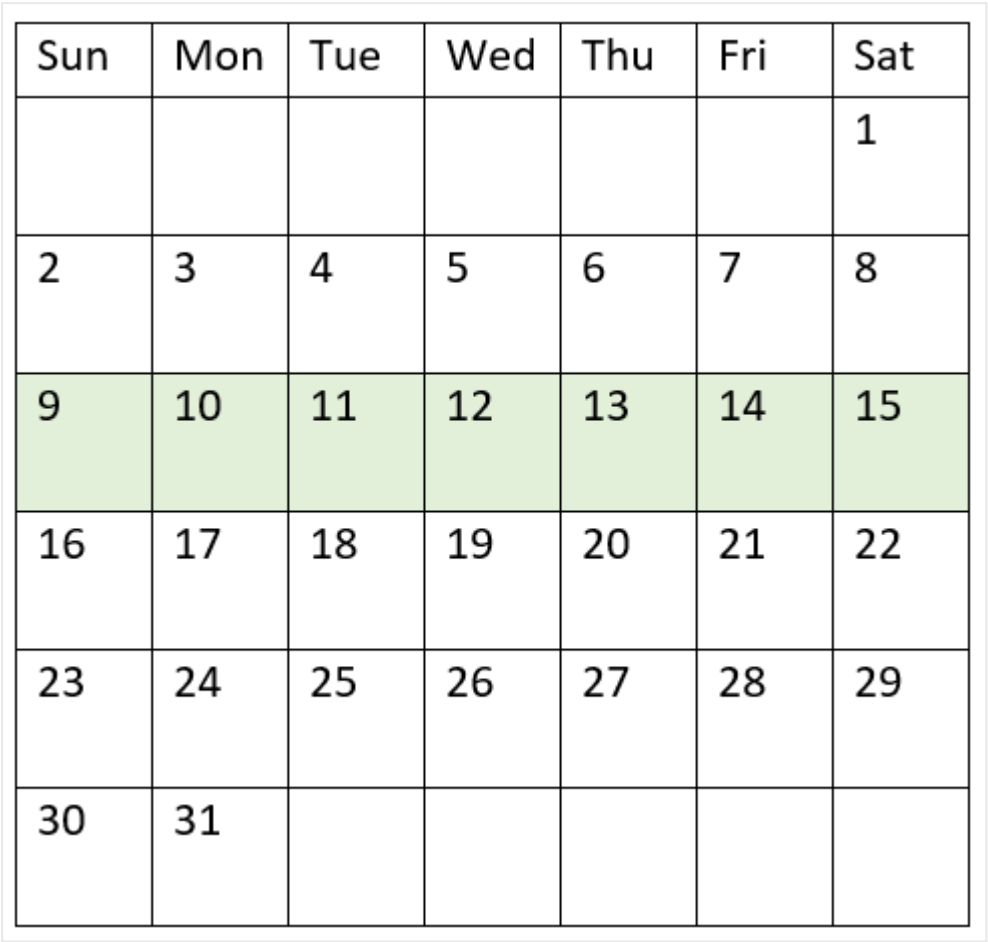
date	week_day	in_week
01/02/2022	So	0
01/05/2022	Mi	0
01/06/2022	Do	0
01/08/2022	Sa	0

date	week_day	in_week
01/09/2022	So	-1
01/10/2022	Mo	-1
01/11/2022	Di	-1
01/12/2022	Mi	-1
01/13/2022	Do	-1
01/14/2022	Fr	-1
01/15/2022	Sa	-1
01/16/2022	So	0
01/17/2022	Mo	0
01/18/2022	Di	0
01/26/2022	Mi	0
01/27/2022	Do	0
01/28/2022	Fr	0
01/29/2022	Sa	0
01/30/2022	So	0
01/31/2022	Mo	0

Das Feld „in_week“ wird in der vorangehenden load-Anweisung mithilfe der Funktion `inweek()` erstellt. Das erste Argument identifiziert, welches Feld ausgewertet wird. Das zweite Argument ist ein hartcodiertes Datum, der 14. Januar. Dies ist das `base_date`. Das Argument `base_date` arbeitet mit der Systemvariablen `FirstWeekDay` zusammen, um die Vergleichswoche zu identifizieren. Eine `period_no` von `0` – was bedeutet, dass die Funktion keine Wochen vor oder nach der segmentierten Woche vergleicht – ist das abschließende Argument.

Die Systemvariable `FirstWeekDay` bestimmt, dass Wochen an einem Sonntag beginnen und an einem Samstag enden. Daher wird der Januar entsprechend dem Diagramm unten in Wochen unterteilt, und die Tage zwischen dem 9. und 15. Januar stellen den gültigen Zeitraum für die Berechnung von `inweek()` dar:

Diagramm eines Kalenders, in dem der Bereich der Funktion `inweek()` hervorgehoben ist.



Sun	Mon	Tue	Wed	Thu	Fri	Sat
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Jede Transaktion, die zwischen dem 9. und 15. Januar stattfindet, gibt ein boolesches Ergebnis von TRUE zurück.

Beispiel 2 – period_no

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Der gleiche Datensatz, der eine Reihe von Transaktionen für 2022 enthält, wird in eine Tabelle namens „`Transactions`“ geladen.
- Systemvariable `FirstWeekDay`, die auf 6 (Sonntag) festgelegt ist
- Eine vorangehende load-Anweisung, die Folgendes enthält:
    - Die Funktion `inweek ()`, die als das Feld „`prev_week`“ festgelegt ist, das bestimmt, welche Transaktionen eine ganze Woche vor der Woche des 14. Januar 2022 stattfanden.

- Die Funktion weekday(), die als das Feld „week_day“ festgelegt ist, das zeigt, welcher Tag der Woche jedem Datum entspricht.

Ladeskript

```
SET FirstWeekDay=6;
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
       *,
       weekday(date) as week_day,
       inweek(date,'01/14/2022', -1) as prev_week
       ;
Load
*
Inline
[
id,date,amount
8188,'01/02/2022',37.23
8189,'01/05/2022',17.17
8190,'01/06/2022',88.27
8191,'01/08/2022',57.42
8192,'01/09/2022',53.80
8193,'01/10/2022',82.06
8194,'01/11/2022',40.39
8195,'01/12/2022',87.21
8196,'01/13/2022',95.93
8197,'01/14/2022',45.89
8198,'01/15/2022',36.23
8199,'01/16/2022',25.66
8200,'01/17/2022',82.77
8201,'01/18/2022',69.98
8202,'01/26/2022',76.11
8203,'01/27/2022',25.12
8204,'01/28/2022',46.23
8205,'01/29/2022',84.21
8206,'01/30/2022',96.24
8207,'01/31/2022',67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

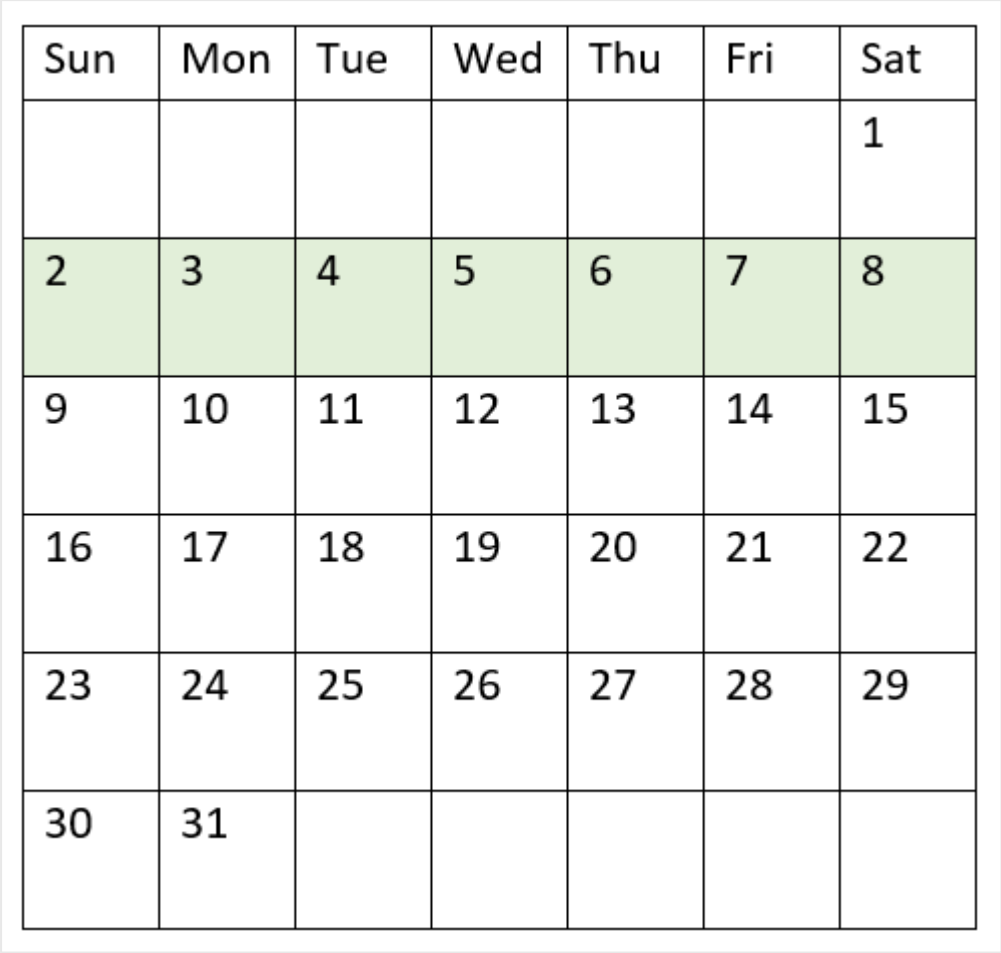
- date
- week_day
- prev_week

Ergebnistabelle

date	week_day	prev_week
01/02/2022	So	-1
01/05/2022	Mi	-1
01/06/2022	Do	-1
01/08/2022	Sa	-1
01/09/2022	So	0
01/10/2022	Mo	0
01/11/2022	Di	0
01/12/2022	Mi	0
01/13/2022	Do	0
01/14/2022	Fr	0
01/15/2022	Sa	0
01/16/2022	So	0
01/17/2022	Mo	0
01/18/2022	Di	0
01/26/2022	Mi	0
01/27/2022	Do	0
01/28/2022	Fr	0
01/29/2022	Sa	0
01/30/2022	So	0
01/31/2022	Mo	0

Da `-1` als Argument `period_no` in der Funktion `inweek()` verwendet wird, werden die Grenzen der Vergleichswoche um volle sieben Tage zurück verschoben. Mit einer `period_no` von `0` liegt die Woche zwischen dem 9. und 15. Januar. In diesem Beispiel verschiebt aber die `period_no` von `-1` die Start- und Endgrenze dieses Segments um eine Woche zurück. Die Datumsgrenzen sind jetzt der 2. bis 8. Januar.

Diagramm eines Kalenders, in dem der Bereich der Funktion `inweek()` hervorgehoben ist.



Sun	Mon	Tue	Wed	Thu	Fri	Sat
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Daher gibt jede Transaktion, die zwischen dem 2. und 8. Januar stattfindet, ein boolesches Ergebnis von TRUE zurück.

Beispiel 3 – first_week_day

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Der gleiche Datensatz, der eine Reihe von Transaktionen für 2022 enthält, wird in eine Tabelle namens „`Transactions`“ geladen.
- Systemvariable `FirstWeekDay`, die auf 6 (Sonntag) festgelegt ist
- Eine vorangehende load-Anweisung, die Folgendes enthält:
    - Die Funktion `inweek()`, die als das Feld „`in_week`“ festgelegt ist, das bestimmt, welche Transaktionen in der Woche vom 14. Januar 2022 stattfanden.

- Die Funktion weekday(), die als das Feld „week_day“ festgelegt ist, das zeigt, welcher Tag der Woche jedem Datum entspricht.

Ladeskript

```
SET FirstWeekDay=6;
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
       *,
       weekday(date) as week_day,
       inweek(date,'01/14/2022', 0, 0) as in_week
       ;
Load
*
Inline
[
id,date,amount
8188,'01/02/2022',37.23
8189,'01/05/2022',17.17
8190,'01/06/2022',88.27
8191,'01/08/2022',57.42
8192,'01/09/2022',53.80
8193,'01/10/2022',82.06
8194,'01/11/2022',40.39
8195,'01/12/2022',87.21
8196,'01/13/2022',95.93
8197,'01/14/2022',45.89
8198,'01/15/2022',36.23
8199,'01/16/2022',25.66
8200,'01/17/2022',82.77
8201,'01/18/2022',69.98
8202,'01/26/2022',76.11
8203,'01/27/2022',25.12
8204,'01/28/2022',46.23
8205,'01/29/2022',84.21
8206,'01/30/2022',96.24
8207,'01/31/2022',67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- date
- week_day
- in_week

Ergebnistabelle

date	week_day	in_week
01/02/2022	So	0
01/05/2022	Mi	0
01/06/2022	Do	0
01/08/2022	Sa	0
01/09/2022	So	0
01/10/2022	Mo	-1
01/11/2022	Di	-1
01/12/2022	Mi	-1
01/13/2022	Do	-1
01/14/2022	Fr	-1
01/15/2022	Sa	-1
01/16/2022	So	-1
01/17/2022	Mo	0
01/18/2022	Di	0
01/26/2022	Mi	0
01/27/2022	Do	0
01/28/2022	Fr	0
01/29/2022	Sa	0
01/30/2022	So	0
01/31/2022	Mo	0

Da `0` als Argument `first_week_day` in der Funktion `inweek()` verwendet wird, wird die Systemvariable `FirstWeekDay` überschrieben und Montag als der erste Tag der Woche festgelegt.

Diagramm eines Kalenders, in dem der Bereich der Funktion `inweek()` hervorgehoben ist.

Sun	Mon	Tue	Wed	Thu	Fri	Sat
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Daher gibt jede Transaktion, die zwischen dem 10. und 16. Januar stattfindet, ein boolesches Ergebnis von TRUE zurück.

Beispiel 4 – Diagrammobjektbeispiel

Ladeskript und Diagrammformel

Übersicht

Es werden derselbe Datensatz und dasselbe Szenario wie im ersten Beispiel verwendet.

In diesem Beispiel wird jedoch der unveränderte Datensatz in die Anwendung geladen. Erstellen Sie eine Kennzahl in der Ergebnistabelle, um zu bestimmen, welche Transaktionen in der Woche vom 14. Januar 2022 stattfanden.

Ladeskript

```
SET FirstWeekDay=6;
SET DateFormat='MM/DD/YYYY';

Transactions:
```

```
Load
*
Inline
[
id,date,amount
8188,'01/02/2022',37.23
8189,'01/05/2022',17.17
8190,'01/06/2022',88.27
8191,'01/08/2022',57.42
8192,'01/09/2022',53.80
8193,'01/10/2022',82.06
8194,'01/11/2022',40.39
8195,'01/12/2022',87.21
8196,'01/13/2022',95.93
8197,'01/14/2022',45.89
8198,'01/15/2022',36.23
8199,'01/16/2022',25.66
8200,'01/17/2022',82.77
8201,'01/18/2022',69.98
8202,'01/26/2022',76.11
8203,'01/27/2022',25.12
8204,'01/28/2022',46.23
8205,'01/29/2022',84.21
8206,'01/30/2022',96.24
8207,'01/31/2022',67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu:

- `date`

Erstellen Sie die folgenden Kennzahlen:

- `=inweek (date,'01/14/2022',0)`, um zu berechnen, ob Transaktionen in der Woche vom 14. Januar stattfanden.
- `=weekday(date)`, um zu zeigen, welcher Tag der Woche jeweils dem Datum entspricht.

Ergebnistabelle

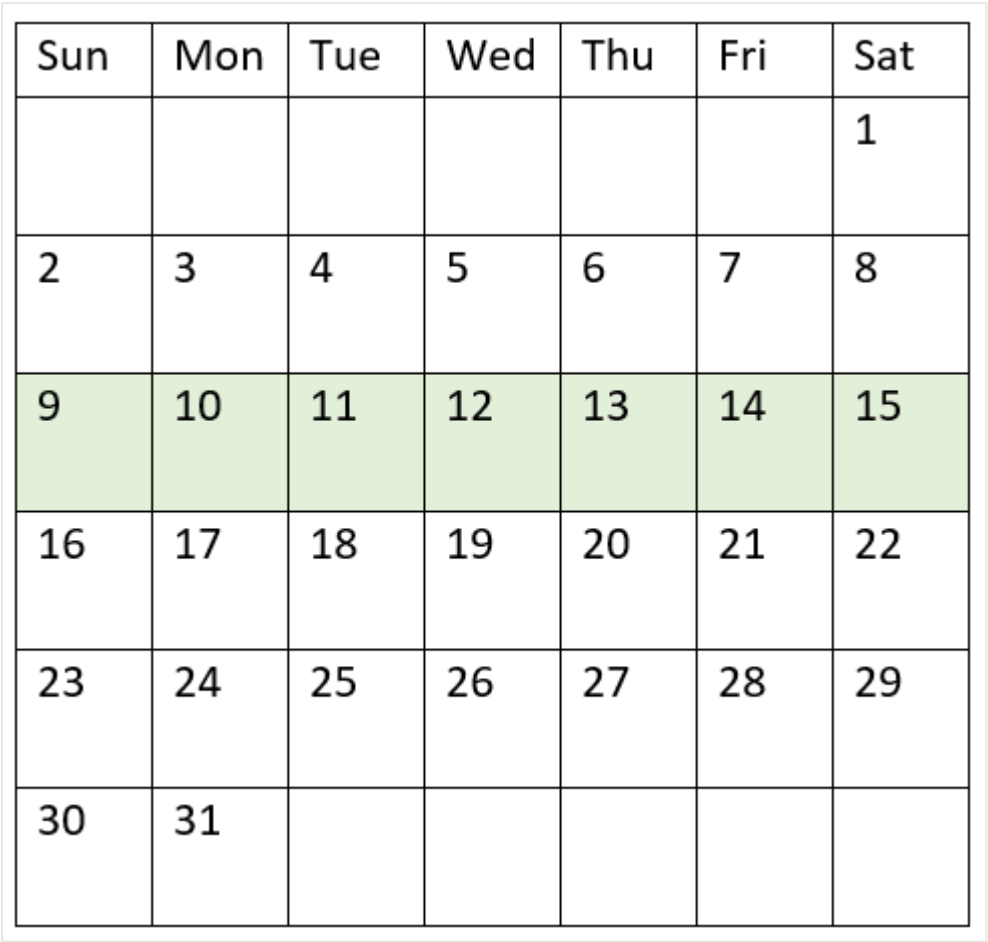
date	week_day	=inweek (date,'01/14/2022',0)
01/02/2022	So	0
01/05/2022	Mi	0
01/06/2022	Do	0
01/08/2022	Sa	0
01/09/2022	So	-1

date	week_day	=inweek (date,'01/14/2022',0)
01/10/2022	Mo	-1
01/11/2022	Di	-1
01/12/2022	Mi	-1
01/13/2022	Do	-1
01/14/2022	Fr	-1
01/15/2022	Sa	-1
01/16/2022	So	0
01/17/2022	Mo	0
01/18/2022	Di	0
01/26/2022	Mi	0
01/27/2022	Do	0
01/28/2022	Fr	0
01/29/2022	Sa	0
01/30/2022	So	0
01/31/2022	Mo	0

Die Kennzahl „in_week“ wird im Diagramm anhand der Funktion `inweek()` erstellt. Das erste Argument identifiziert, welches Feld ausgewertet wird. Das zweite Argument ist ein hartcodiertes Datum, der 14. Januar. Dies ist das `base_date`. Das Argument `base_date` arbeitet mit der Systemvariablen `FirstWeekDay` zusammen, um die Vergleichswoche zu identifizieren. Eine `period_no` von `0` ist das abschließende Argument.

Die Systemvariable `FirstWeekDay` bestimmt, dass Wochen an einem Sonntag beginnen und an einem Samstag enden. Daher wird der Januar entsprechend dem Diagramm unten in Wochen unterteilt, und die Tage zwischen dem 9. und 15. Januar stellen den gültigen Zeitraum für die Berechnung von `inweek()` dar:

Diagramm eines Kalenders, in dem der Bereich der Funktion `inweek()` hervorgehoben ist.



Sun	Mon	Tue	Wed	Thu	Fri	Sat
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Jede Transaktion, die zwischen dem 9. und 15. Januar stattfindet, gibt ein boolesches Ergebnis von TRUE zurück.

Beispiel 5 – Szenario

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Datensatz, der in eine Tabelle namens „`Products`“ geladen wird
- Die Tabelle enthält die folgenden Felder:
    - product ID
    - product type
    - manufacture date
    - cost price

Es wurde festgestellt, dass aufgrund eines Gerätefehlers die in der Woche vom 12. Januar hergestellten Produkte mangelhaft waren. Der Endbenutzer möchte ein Diagramm, das nach Woche den Status der hergestellten Produkte angibt und zeigt, welche „mangelhaft“ und welche „einwandfrei“ waren und was die in dieser Woche gefertigten Produkte gekostet haben.

Ladeskript

```
Products:
Load
*
Inline
[
product_id,manufacture_date,cost_price
8188,'01/02/2022',37.23
8189,'01/05/2022',17.17
8190,'01/06/2022',88.27
8191,'01/08/2022',57.42
8192,'01/09/2022',53.80
8193,'01/10/2022',82.06
8194,'01/11/2022',40.39
8195,'01/12/2022',87.21
8196,'01/13/2022',95.93
8197,'01/14/2022',45.89
8198,'01/15/2022',36.23
8199,'01/16/2022',25.66
8200,'01/17/2022',82.77
8201,'01/18/2022',69.98
8202,'01/26/2022',76.11
8203,'01/27/2022',25.12
8204,'01/28/2022',46.23
8205,'01/29/2022',84.21
8206,'01/30/2022',96.24
8207,'01/31/2022',67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu:

- `=weekname(manufacture_date)`

Erstellen Sie die folgenden Kennzahlen:

- `=if(only(inweek(manufacture_date,makedate(2022,01,12),0)),'Defective','Faultless')`, um zu identifizieren, welche Produkte mangelhaft und welche einwandfrei sind. Verwenden Sie dafür die Funktion `inweek()`.
- `=sum(cost_price)`, um die Summe der Kosten für jedes Produkt zu zeigen.

Gehen Sie folgendermaßen vor:

1. Legen Sie das **Zahlenformat** der Kennzahl auf **Währung** fest.
2. Deaktivieren Sie unter **Darstellung** die Option **Gesamtwerte**.

Ergebnistabelle

weekname (manufacture_date)	=if(only(inweek(manufacture_date,makedate(2022,01,12),0)), 'Defective','Faultless')	Sum(cost_ price)
2022/02	Faultless	200.09
2022/03	Defective	441.51
2022/04	Faultless	178.41
2022/05	Faultless	231.67
2022/06	Faultless	163.91

Die Funktion `inweek()` gibt einen booleschen Wert zurück, wenn sie das Herstellungsdatum der einzelnen Produkte auswertet. Für alle in der Woche vom 12. Januar hergestellten Produkte gibt die Funktion `inweek()` einen booleschen Wert von TRUE zurück und markiert die Produkte als „mangelhaft“. Alle Produkte, die einen Wert von FALSE zurückgeben und daher nicht in der betreffenden Woche gefertigt wurden, werden als „einwandfrei“ markiert.

inweektodate

Diese Funktion liefert True, wenn **timestamp** innerhalb des Teils der Woche liegt, der **base_date** enthält, und zwar bis einschließlich der letzten Millisekunde von **base_date**.

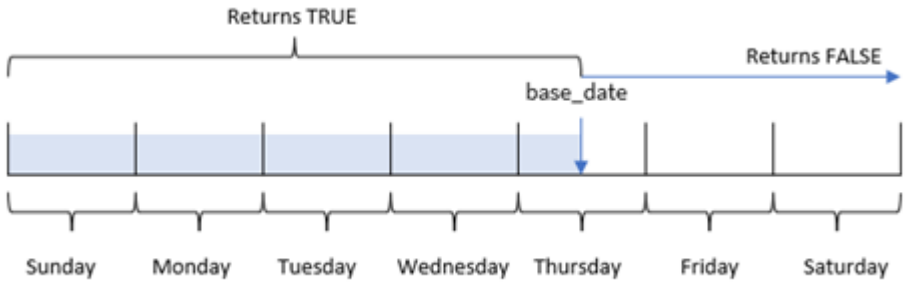
Syntax:

```
InWeekToDate (timestamp, base_date, period_no [, first_week_day])
```

Rückgabe Datentyp: Boolesch

> *In Qlik Sense wird der boolesche Wert „wahr“ durch -1 dargestellt, der Wert „falsch“ durch 0.*

Diagramm der Funktion „inweektodate“



Die Funktion `inweektodate()` verwendet den Parameter `base_date` zum Identifizieren eines maximalen Grenzdatums eines Wochensegments sowie des entsprechenden Datums für den Wochenbeginn, das auf der Systemvariablen `FirstWeekDay` basiert (oder dem benutzerdefinierten Parameter `first_week_day`). Nachdem dieses Wochensegment definiert wurde, gibt die Funktion boolesche Ergebnisse zurück, wenn die vordefinierten Datumswerte mit diesem Segment verglichen werden.

Verwendung

Die Funktion `inweektodate()` gibt einen booleschen Wert zurück. In der Regel wird dieser Funktionstyp als Bedingung in einem `if`-Ausdruck verwendet. Damit wird eine Aggregierung oder Berechnung zurückgegeben, abhängig davon, ob ein ausgewertetes Datum in die betreffende Woche bis einschließlich eines bestimmten Datums fällt.

Beispielsweise kann die Funktion `inweektodate()` verwendet werden, um alle Umsätze zu berechnen, die in einer angegebenen Woche bis zu einem bestimmten Datum getätigt wurden.

Argumente

Argument	Beschreibung
timestamp	Das Datum, das mit **base_date** verglichen werden soll.
base_date	Datum, das für die Interpretation der Woche verwendet wird.
period_no	Mit **period_no** kann der Beginn der Woche festgelegt werden. **period_no** ist eine ganze Zahl, wobei 0 für die Woche steht, die **base_date** enthält. Negative Werte von **period_no** stehen für vorangehende Wochen, positive Werte für nachfolgende Wochen.
first_week_day	Standardmäßig ist der erste Tag der Woche der Sonntag (wie durch die Systemvariable „FirstWeekDay“ definiert) und beginnt um Mitternacht zwischen Samstag und Sonntag. Der Parameter **first_week_day** überschreibt die Variable **FirstWeekDay**. Wenn der Wochenbeginn auf einen anderen Tag fallen soll, geben Sie ein Flag zwischen 0 und 6 an. Für eine Woche, die am Montag beginnt und am Sonntag endet, verwenden Sie das Flag 0 für Montag, 1 für Dienstag, 2 für Mittwoch, 3 für Donnerstag, 4 für Freitag, 5 für Samstag und 6 für Sonntag.

Funktionsbeispiele

Beispiel	Interaktion
`inweektodate ('01/12/2006', '01/12/2006', 0)`	Liefert TRUE.
`inweektodate ('01/12/2006', '01/11/2006', 0)`	Gibt FALSE zurück.

Beispiel	Interaktion
`inweektodate ('01/12/2006', '01/18/2006', -1)`	Liefert FALSE. Weil `period_no` als -1 festgelegt wurde, wird als Gültigkeitsdatum, mit dem `timestamp` verglichen wird, 01/11/2006 verwendet.
`inweektodate ('01/11/2006', '01/12/2006', 0, 3 )`	Gibt FALSE zurück, weil `first_week_day` als 3 (Donnerstag) festgelegt wurde, wodurch 01/12/2006 der erste Tag der Folgewoche nach der Woche ist, die 01/12/2006 enthält.

Die folgenden Themen können Sie bei der Arbeit mit dieser Funktion unterstützen:

Verwandte Themen

Thema	Standardkennzeichen/Wert	Beschreibung
FirstWeekDay (page 233)	6 / Sonntag	Definiert den Starttag jeder Woche.

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Beispiel 1 – keine zusätzlichen Argumente

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Datensatz mit einer eine Reihe von Transaktionen für den Monat Januar 2022, der in eine Tabelle namens „`Transactions`“ geladen wird
- Das bereitgestellte Datumsfeld im Format `TimestampFormat='M/D/YYYY h:mm:ss[.fff]'`.
- Die Erstellung eines Felds `in_week_to_date`, das bestimmt, welche Transaktionen in der Woche bis zum 14. Januar 2022 stattgefunden haben.

- Die Erstellung eines zusätzlichen Felds mit dem Namen `weekday` unter Verwendung der Funktion `weekday()`. Dieses neue Feld wird erstellt, um zu zeigen, welcher Tag der Woche jeweils dem Datum entspricht.

Ladeskript

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff]';
SET FirstWeekDay=6;
Transactions:
    Load
        *,
        weekday(date) as week_day,
        inweektodate(date,'01/14/2022', 0) as in_week_to_date
        ;
Load
*
Inline
[
id,date,amount
8188,'2022-01-02 12:22:06',37.23
8189,'2022-01-05 01:02:30',17.17
8190,'2022-01-06 15:36:20',88.27
8191,'2022-01-08 10:58:35',57.42
8192,'2022-01-09 08:53:32',53.80
8193,'2022-01-10 21:13:01',82.06
8194,'2022-01-11 00:57:13',40.39
8195,'2022-01-12 09:26:02',87.21
8196,'2022-01-13 15:05:09',95.93
8197,'2022-01-14 18:44:57',45.89
8198,'2022-01-15 06:10:46',36.23
8199,'2022-01-16 06:39:27',25.66
8200,'2022-01-17 10:44:16',82.77
8201,'2022-01-18 18:48:17',69.98
8202,'2022-01-26 04:36:03',76.11
8203,'2022-01-27 08:07:49',25.12
8204,'2022-01-28 12:24:29',46.23
8205,'2022-01-30 11:56:56',84.21
8206,'2022-01-30 14:40:19',96.24
8207,'2022-01-31 05:28:21',67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `week_day`
- `in_week_to_date`

Ergebnistabelle

date	week_day	in_week_to_date
2022-01-02 12:22:06	So	0
2022-01-05 01:02:30	Mi	0
2022-01-06 15:36:20	Do	0
2022-01-08 10:58:35	Sa	0
2022-01-09 08:53:32	So	-1
2022-01-10 21:13:01	Mo	-1
2022-01-11 00:57:13	Di	-1
2022-01-12 09:26:02	Mi	-1
2022-01-13 15:05:09	Do	-1
2022-01-14 18:44:57	Fr	-1
2022-01-15 06:10:46	Sa	0
2022-01-16 06:39:27	So	0
2022-01-17 10:44:16	Mo	0
2022-01-18 18:48:17	Di	0
2022-01-26 04:36:03	Mi	0
2022-01-27 08:07:49	Do	0
2022-01-28 12:24:29	Fr	0
2022-01-30 11:56:56	So	0
2022-01-30 14:40:19	So	0
2022-01-31 05:28:21	Mo	0

Das Feld „in_week_to_date“ wird im vorangehenden load-Befehl mithilfe der Funktion `inweektodate()` erstellt. Das erste angegebene Argument identifiziert, welches Feld ausgewertet wird. Das zweite Argument ist ein hardcodiertes Datum für den 14. Januar. Dies ist das `base_date`, das identifiziert, welche Woche segmentiert wird, und die Endbegrenzung dieses Segments definiert. Eine `period_no` von 0 ist das abschließende Argument, was bedeutet, dass die Funktion keine Wochen vor oder nach der segmentierten Woche vergleicht.

Die Systemvariable `FirstWeekDay` bestimmt, dass Wochen an einem Sonntag beginnen und an einem Samstag enden. Daher wird der Januar entsprechend dem Diagramm unten in Wochen unterteilt, und die Tage zwischen dem 9. und 14. Januar stellen den gültigen Zeitraum für die Berechnung von `inweekdodate()` dar:

Das Kalenderdiagramm zeigt Transaktionsdaten, die ein boolesches Ergebnis von TRUE zurückgeben

Sun	Mon	Tue	Wed	Thur	Fri	Sat
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Jede Transaktion, die zwischen dem 9. und 14. Januar stattfindet, gibt ein boolesches Ergebnis von TRUE zurück. Transaktionen vor und nach diesen Tagen geben ein boolesches Ergebnis von FALSE zurück.

Beispiel 2 – period_no

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Derselbe Datensatz und dasselbe Szenario wie im ersten Beispiel.
- Die Erstellung eines Felds `prev_week_to_date`, das bestimmt, welche Transaktionen in einer vollständigen Woche vor dem Wochensegment stattfanden, das am 14. Januar 2022 endet.
- Die Erstellung eines zusätzlichen Felds mit dem Namen `weekday` unter Verwendung der Funktion `weekday()`. Damit soll gezeigt werden, welcher Tag der Woche jeweils dem Datum entspricht.

Ladeskript

```
SET FirstWeekDay=6;
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff]';
Transactions:
    Load
        *,
        weekday(date) as week_day,
        inweektodate(date,'01/14/2022', -1) as prev_week_to_date
        ;
Load
*
Inline
[
id,date,amount
```

```
8188,'2022-01-02 12:22:06',37.23
8189,'2022-01-05 01:02:30',17.17
8190,'2022-01-06 15:36:20',88.27
8191,'2022-01-08 10:58:35',57.42
8192,'2022-01-09 08:53:32',53.80
8193,'2022-01-10 21:13:01',82.06
8194,'2022-01-11 00:57:13',40.39
8195,'2022-01-12 09:26:02',87.21
8196,'2022-01-13 15:05:09',95.93
8197,'2022-01-14 18:44:57',45.89
8198,'2022-01-15 06:10:46',36.23
8199,'2022-01-16 06:39:27',25.66
8200,'2022-01-17 10:44:16',82.77
8201,'2022-01-18 18:48:17',69.98
8202,'2022-01-26 04:36:03',76.11
8203,'2022-01-27 08:07:49',25.12
8204,'2022-01-28 12:24:29',46.23
8205,'2022-01-30 11:56:56',84.21
8206,'2022-01-30 14:40:19',96.24
8207,'2022-01-31 05:28:21',67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `week_day`
- `prev_week_to_date`

Ergebnistabelle

date	week_day	prev_week_to_date
2022-01-02 12:22:06	So	-1
2022-01-05 01:02:30	Mi	-1
2022-01-06 15:36:20	Do	-1
2022-01-08 10:58:35	Sa	0
2022-01-09 08:53:32	So	0
2022-01-10 21:13:01	Mo	0
2022-01-11 00:57:13	Di	0
2022-01-12 09:26:02	Mi	0
2022-01-13 15:05:09	Do	0
2022-01-14 18:44:57	Fr	0

date	week_day	prev_week_to_date
2022-01-15 06:10:46	Sa	0
2022-01-16 06:39:27	So	0
2022-01-17 10:44:16	Mo	0
2022-01-18 18:48:17	Di	0
2022-01-26 04:36:03	Mi	0
2022-01-27 08:07:49	Do	0
2022-01-28 12:24:29	Fr	0
2022-01-30 11:56:56	So	0
2022-01-30 14:40:19	So	0
2022-01-31 05:28:21	Mo	0

Ein `period_no`-Wert von -1 gibt an, dass die Funktion `inweektodate ()` das Eingabequartalssegment mit der vorherigen Woche vergleicht. Das Wochensegment entspricht anfänglich den Tagen vom 9. bis 14. Januar. Die `period_no` verschiebt dann sowohl die Start- als auch die Endgrenze dieses Segments auf eine Woche früher, sodass die Datumsgrenzen der 2. bis 7. Januar werden.

Das Kalenderdiagramm zeigt Transaktionsdaten, die ein boolesches Ergebnis von TRUE zurückgeben

Sun	Mon	Tue	Wed	Thu	Fri	Sat
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Daher gibt jede Transaktion, die zwischen dem 2. Januar und 8. Januar (ausschließlich des 8. Januar selbst) stattfindet, ein boolesches Ergebnis von TRUE zurück.

Beispiel 3 – first_week_day

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Derselbe Datensatz und dasselbe Szenario wie im ersten Beispiel.
- Die Erstellung eines Felds `in_week_to_date`, das bestimmt, welche Transaktionen in der Woche bis zum 14. Januar 2022 stattgefunden haben.
- Die Erstellung eines zusätzlichen Felds mit dem Namen `weekday` unter Verwendung der Funktion `weekday()`. Damit soll gezeigt werden, welcher Tag der Woche jeweils dem Datum entspricht.

In diesem Beispiel wird Montag als der erste Tag der Woche betrachtet.

Ladeskript

```
SET FirstWeekDay=6;
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff]';

Transactions:
    Load
        *,
        weekday(date) as week_day,
        inweektodate(date,'01/14/2022', 0, 0) as in_week_to_date
        ;
Load
*
Inline
[
id,date,amount
8188,'2022-01-02 12:22:06',37.23
8189,'2022-01-05 01:02:30',17.17
8190,'2022-01-06 15:36:20',88.27
8191,'2022-01-08 10:58:35',57.42
8192,'2022-01-09 08:53:32',53.80
8193,'2022-01-10 21:13:01',82.06
8194,'2022-01-11 00:57:13',40.39
8195,'2022-01-12 09:26:02',87.21
8196,'2022-01-13 15:05:09',95.93
8197,'2022-01-14 18:44:57',45.89
8198,'2022-01-15 06:10:46',36.23
8199,'2022-01-16 06:39:27',25.66
8200,'2022-01-17 10:44:16',82.77
8201,'2022-01-18 18:48:17',69.98
8202,'2022-01-26 04:36:03',76.11
8203,'2022-01-27 08:07:49',25.12
8204,'2022-01-28 12:24:29',46.23
8205,'2022-01-30 11:56:56',84.21
8206,'2022-01-30 14:40:19',96.24
8207,'2022-01-31 05:28:21',67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- date
- week_day
- in_week_to_date

Ergebnistabelle

date	week_day	in_week_to_date
2022-01-02 12:22:06	So	0
2022-01-05 01:02:30	Mi	0
2022-01-06 15:36:20	Do	0
2022-01-08 10:58:35	Sa	0
2022-01-09 08:53:32	So	0
2022-01-10 21:13:01	Mo	-1
2022-01-11 00:57:13	Di	-1
2022-01-12 09:26:02	Mi	-1
2022-01-13 15:05:09	Do	-1
2022-01-14 18:44:57	Fr	-1
2022-01-15 06:10:46	Sa	0
2022-01-16 06:39:27	So	0
2022-01-17 10:44:16	Mo	0
2022-01-18 18:48:17	Di	0
2022-01-26 04:36:03	Mi	0
2022-01-27 08:07:49	Do	0
2022-01-28 12:24:29	Fr	0
2022-01-30 11:56:56	So	0
2022-01-30 14:40:19	So	0
2022-01-31 05:28:21	Mo	0

Indem 0 als Argument `first_week_day` in der Funktion `inweektodate()` verwendet wird, überschreibt das Funktionsargument die Systemvariable `FirstWeekDay` und legt Montag als den ersten Tag der Woche fest.

Das Kalenderdiagramm zeigt Transaktionsdaten, die ein boolesches Ergebnis von TRUE zurückgeben

Mon	Tue	Wed	Thu	Fri	Sat	Sun
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	17
24	25	26	27	28	29	30
31						

Daher gibt jede Transaktion, die zwischen dem 10. und 14. Januar stattfindet, ein boolesches Ergebnis von TRUE zurück, während Transaktionen mit Datum außerhalb dieser Grenzen einen Wert von FALSE zurückgeben.

Beispiel 4 – Diagrammobjektbeispiel

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript verwendet den gleichen Datensatz und das gleiche Szenario wie das erste Beispiel. In diesem Beispiel wird jedoch der unveränderte Datensatz in die Anwendung geladen. Die Berechnung, die bestimmt, welche Transaktionen in der Woche bis zum 14. Januar 2022 stattfanden, wird als Kennzahl in einem Diagrammobjekt erstellt.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8188,'2022-01-02 12:22:06',37.23
8189,'2022-01-05 01:02:30',17.17
8190,'2022-01-06 15:36:20',88.27
8191,'2022-01-08 10:58:35',57.42
8192,'2022-01-09 08:53:32',53.80
8193,'2022-01-10 21:13:01',82.06
8194,'2022-01-11 00:57:13',40.39
```

```
8195,'2022-01-12 09:26:02',87.21
8196,'2022-01-13 15:05:09',95.93
8197,'2022-01-14 18:44:57',45.89
8198,'2022-01-15 06:10:46',36.23
8199,'2022-01-16 06:39:27',25.66
8200,'2022-01-17 10:44:16',82.77
8201,'2022-01-18 18:48:17',69.98
8202,'2022-01-26 04:36:03',76.11
8203,'2022-01-27 08:07:49',25.12
8204,'2022-01-28 12:24:29',46.23
8205,'2022-01-30 11:56:56',84.21
8206,'2022-01-30 14:40:19',96.24
8207,'2022-01-31 05:28:21',67.67
];
```

Ergebnisse

Gehen Sie folgendermaßen vor:

1. Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu: `date`.
2. Um zu berechnen, ob Transaktionen in der gleichen Woche bis zum 14. Januar stattfanden, erstellen Sie die folgende Kennzahl:
   `=inweektodate(date,'01/14/2022',0)`
3. Um zu zeigen, welcher Tag der Woche jeweils dem Datum entspricht, erstellen Sie eine weitere Kennzahl:
   `=weekday(date)`

Ergebnistabelle

date	week_day	in_week_to_date
2022-01-02 12:22:06	So	0
2022-01-05 01:02:30	Mi	0
2022-01-06 15:36:20	Do	0
2022-01-08 10:58:35	Sa	0
2022-01-09 08:53:32	So	-1
2022-01-10 21:13:01	Mo	-1
2022-01-11 00:57:13	Di	-1
2022-01-12 09:26:02	Mi	-1
2022-01-13 15:05:09	Do	-1
2022-01-14 18:44:57	Fr	-1
2022-01-15 06:10:46	Sa	0
2022-01-16 06:39:27	So	0

date	week_day	in_week_to_date
2022-01-17 10:44:16	Mo	0
2022-01-18 18:48:17	Di	0
2022-01-26 04:36:03	Mi	0
2022-01-27 08:07:49	Do	0
2022-01-28 12:24:29	Fr	0
2022-01-30 11:56:56	So	0
2022-01-30 14:40:19	So	0
2022-01-31 05:28:21	Mo	0

Das Feld `in_week_to_date` wird als Kennzahl in einem Diagrammobjekt anhand der Funktion `inweektodate()` erstellt. Das erste angegebene Argument identifiziert, welches Feld ausgewertet wird. Das zweite Argument ist ein hardcodiertes Datum für den 14. Januar. Dies ist das `base_date`, das identifiziert, welche Woche segmentiert wird, und die Endbegrenzung dieses Segments definiert. Eine `period_no` von 0 ist das abschließende Argument, was bedeutet, dass die Funktion keine Wochen vor oder nach der segmentierten Woche vergleicht.

Die Systemvariable `FirstWeekDay` bestimmt, dass Wochen an einem Sonntag beginnen und an einem Samstag enden. Daher wird der Januar entsprechend dem Diagramm unten in Wochen unterteilt, und die Tage zwischen dem 9. und 14. Januar stellen den gültigen Zeitraum für die Berechnung von `inweekdodate()` dar:

Das Kalenderdiagramm zeigt Transaktionsdaten, die ein boolesches Ergebnis von TRUE zurückgeben

Sun	Mon	Tue	Wed	Thu	Fri	Sat
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Jede Transaktion, die zwischen dem 9. und 14. Januar stattfindet, gibt ein boolesches Ergebnis von TRUE zurück. Transaktionen vor und nach diesen Tagen geben ein boolesches Ergebnis von FALSE zurück.

Beispiel 5 – Szenario

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz, der in eine Tabelle namens `Products` geladen wird.
- Informationen bezüglich Produkt-ID, Herstellungsdatum und Selbstkosten.

Es wurde festgestellt, dass aufgrund eines Gerätefehlers die in der Woche vom 12. Januar hergestellten Produkte mangelhaft waren. Das Problem wurde am 13. Januar behoben. Der Endbenutzer möchte ein Diagrammobjekt haben, das nach Woche den Status der hergestellten Produkte angibt und zeigt, welche „mangelhaft“ und welche „einwandfrei“ waren und was die in dieser Woche gefertigten Produkte gekostet haben.

Ladeskript

```
Products:
Load
*
Inline
[
product_id,manufacture_date,cost_price
8188,'2022-01-02 12:22:06',37.23
8189,'2022-01-05 01:02:30',17.17
8190,'2022-01-06 15:36:20',88.27
8191,'2022-01-08 10:58:35',57.42
8192,'2022-01-09 08:53:32',53.80
8193,'2022-01-10 21:13:01',82.06
8194,'2022-01-11 00:57:13',40.39
8195,'2022-01-12 09:26:02',87.21
8196,'2022-01-13 15:05:09',95.93
8197,'2022-01-14 18:44:57',45.89
8198,'2022-01-15 06:10:46',36.23
8199,'2022-01-16 06:39:27',25.66
8200,'2022-01-17 10:44:16',82.77
8201,'2022-01-18 18:48:17',69.98
8202,'2022-01-26 04:36:03',76.11
8203,'2022-01-27 08:07:49',25.12
8204,'2022-01-28 12:24:29',46.23
8205,'2022-01-30 11:56:56',84.21
8206,'2022-01-30 14:40:19',96.24
8207,'2022-01-31 05:28:21',67.67
];
```

Ergebnisse

Gehen Sie folgendermaßen vor:

1. Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle. Erstellen Sie eine Dimension, um die Wochennamen anzuzeigen:
   `=weekname(manufacture_date)`
2. Erstellen Sie dann eine Dimension, um zu identifizieren, welche Produkte mangelhaft und welche einwandfrei sind:
   `=if(inWeektodate(manufacture_date,makedate(2022,01,12),0),'Defective','Faultless')`
3. Erstellen Sie eine Kennzahl zum Summieren des `cost_price` der Produkte:
   `=sum(cost_price)`
4. Legen Sie das **Zahlenformat** der Kennzahl auf **Währung** fest.

Ergebnistabelle

weekname(manufacture_date)	if(inweektodate(manufacture_date,makedate (2022,01,12),0),'Defective','Faultless')	Sum(cost_price)
2022/02	Faultless	$200.09
2022/03	Defective	$263.46
2022/03	Faultless	$178.05
2022/04	Faultless	$178.41
2022/05	Faultless	$147.46
2022/06	Faultless	$248.12

Die Funktion `inweektodate()` gibt einen booleschen Wert zurück, wenn sie das Herstellungsdatum der einzelnen Produkte auswertet. Für diejenigen, die einen booleschen Wert von `TRUE` zurückgeben, markiert sie die Produkte als `'Defective'`. Für jedes Produkt, das einen Wert von `FALSE` zurückgibt und somit nicht in der Woche bis zum 12. Januar hergestellt wurde, werden die Produkte als `'Faultless'` markiert.

inyear

Diese Funktion liefert True, wenn **timestamp** innerhalb des Jahres liegt, das **base_date** enthält.

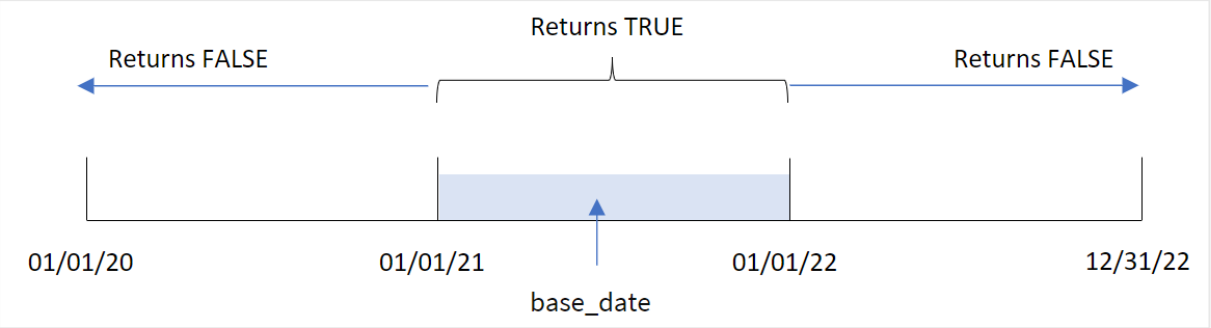
Syntax:

```
InYear (timestamp, base_date, period_no [, first_month_of_year])
```

Rückgabe Datentyp: Boolesch

In Qlik Sense wird der boolesche Wert „wahr“ durch -1 dargestellt, der Wert „falsch“ durch 0.

Diagramm des Bereichs der Funktion `inyear()`



Die Funktion `inyear()` gibt ein boolesches Ergebnis zurück, wenn die ausgewählten Datumswerte mit einem vom `base_date` definierten Jahr verglichen werden.

Verwendung

Die Funktion `inyear()` gibt einen booleschen Wert zurück. In der Regel wird dieser Funktionstyp als Bedingung in einer `if expression` verwendet. Er gibt eine Aggregierung oder Berechnung zurück, abhängig davon, ob ein ausgewertetes Datum im betreffenden Jahr liegt. Beispielsweise kann die Funktion `inyear()` verwendet werden, um alle Umsätze in einem definierten Jahr zu identifizieren.

Argumente

Argument	Beschreibung
timestamp	Das Datum, das mit **base_date** verglichen werden soll.
base_date	Datum, das für die Interpretation des Jahres verwendet wird.
period_no	Mit **period_no** können Sie den Beginn des Jahres festlegen.**period_no** ist eine ganze Zahl, wobei 0 für das Jahr steht, das **base_date** enthält. Negative Werte von **period_no** stehen für vorangehende, positive Werte für nachfolgende Jahre.
first_month_of_year	Wenn Sie mit abweichenden Geschäftsjahren arbeiten möchten, definieren Sie mit einer Zahl zwischen 2 und 12 für **first_month_of_year** einen anderen Startmonat.

Sie können die folgenden Werte verwenden, um den ersten Monat des Jahres im Argument `first_month_of_year` festzulegen.

Werte für first_month_of_year

Monat	Wert
Februar	2
März	3
April	4
Mai	5

Monat	Wert
Juni	6
Juli	7
August	8
September	9
Oktober	10
November	11
Dezember	12

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Funktionsbeispiele

Beispiel	Ergebnis
`inyear ('01/25/2013', '01/01/2013', 0 )`	Gibt TRUE zurück
`inyear ('01/25/2012', '01/01/2013', 0)`	Gibt FALSE zurück
`inyear ('01/25/2013', '01/01/2013', -1)`	Gibt FALSE zurück
`inyear ('01/25/2012', '01/01/2013', -1 )`	Gibt TRUE zurück
`inyear ('01/25/2013', '01/01/2013', 0, 3)`	Gibt TRUE zurück Die Werte von `base_date` und `first_month_of_year` geben an, dass der Zeitstempel zwischen dem 01/03/2012 und dem 02/28/2013 liegen muss.
`inyear ('03/25/2013', '07/01/2013', 0, 3 )`	Gibt TRUE zurück

Beispiel 1 – einfaches Beispiel

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz, der eine Reihe von Transaktionen zwischen 2020 und 2022 enthält, wird in eine Tabelle namens „`Transactions`“ geladen.
- Ein vorangehender load-Befehl, der die Funktion `inyear()` enthält, wird als das Feld „`in_year`“ festgelegt und bestimmt, welche Transaktionen im gleichen Jahr wie der 26. Juli 2021 stattfanden.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';
Transactions:
    Load
        *,
        inyear(date,'07/26/2021', 0) as in_year
        ;
Load
*
Inline
[
id,date,amount
8188,'01/13/2020',37.23
8189,'02/26/2020',17.17
8190,'03/27/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'08/14/2020',82.06
8194,'10/07/2020',40.39
8195,'12/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'12/27/2021',25.12
8204,'06/06/2022',46.23
8205,'07/18/2022',84.21
8206,'11/14/2022',96.24
8207,'12/12/2022',67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

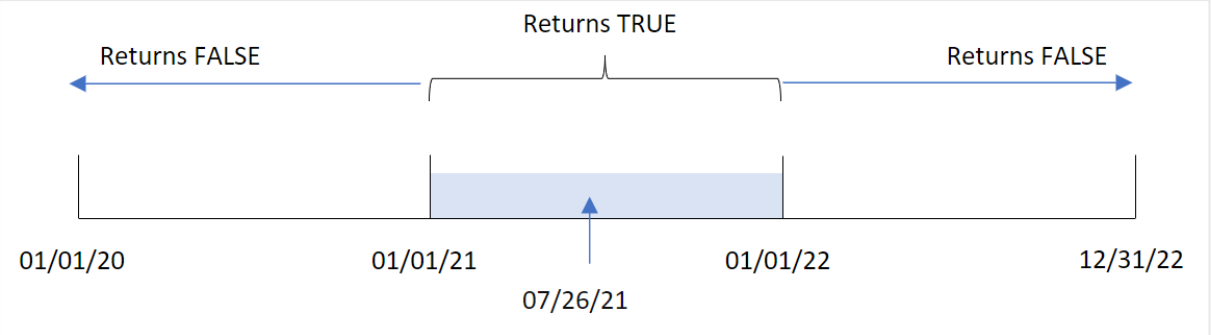
- `date`
- `in_year`

Ergebnistabelle

date	in_year
01/13/2020	0
02/26/2020	0
03/27/2020	0
04/16/2020	0
05/21/2020	0
08/14/2020	0
10/07/2020	0
12/05/2020	0
01/22/2021	-1
02/03/2021	-1
03/17/2021	-1
04/23/2021	-1
05/04/2021	-1
06/30/2021	-1
07/26/2021	-1
12/27/2021	-1
06/06/2022	0
07/18/2022	0
11/14/2022	0
12/12/2022	0

Das Feld „`in_year`“ wird in der vorangehenden load-Anweisung mithilfe der Funktion `inyear()` erstellt. Das erste Argument identifiziert, welches Feld ausgewertet wird. Das zweite Argument ist ein hartcodiertes Datum, der 26. Juli 2021. Dies ist das `base_date`, das das Vergleichsjahr bestimmt. Eine `period_no` von `0` ist das abschließende Argument, was bedeutet, dass die Funktion `inyear()` keine Jahre vor oder nach dem Jahr vergleicht.

Diagramm des Bereichs der Funktion `inyear()` mit dem 26. Juli als Basisdatum.



Jede Transaktion, die im Jahr 2021 stattfindet, gibt ein boolesches Ergebnis von TRUE zurück.

Beispiel 2 – period_no

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Datensatz mit einer Reihe von Transaktionen zwischen 2020 und 2022, der in eine Tabelle namens „`Transactions`“ geladen wird
- Eine vorangehende load-Anweisung, die die Funktion `inyear()` enthält, die als das Feld „`previous_year`“ festgelegt wird und bestimmt, welche Transaktionen im Jahr vor dem Jahr mit dem 26. Juli 2021 stattfanden

Ladeskript

```
SET DateFormat='MM/DD/YYYY';
Transactions:
    Load
        *,
        inyear(date,'07/26/2021', -1) as previous_year
        ;
Load
*
Inline
[
id,date,amount
8188,'01/13/2020',37.23
8189,'02/26/2020',17.17
8190,'03/27/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'08/14/2020',82.06
8194,'10/07/2020',40.39
8195,'12/05/2020',87.21
8196,'01/22/2021',95.93
```

```
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'12/27/2021',25.12
8204,'06/06/2022',46.23
8205,'07/18/2022',84.21
8206,'11/14/2022',96.24
8207,'12/12/2022',67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `previous_year`

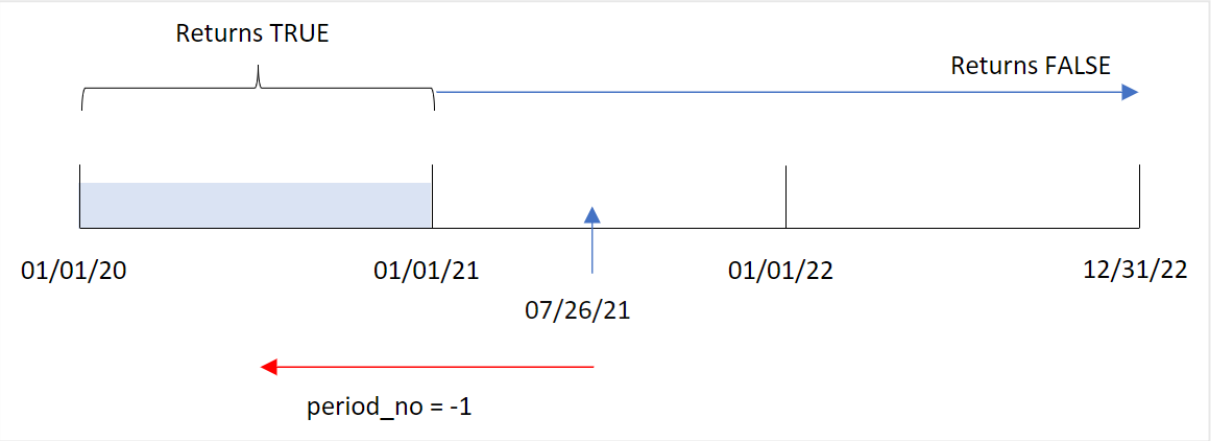
Ergebnistabelle

date	previous_year
01/13/2020	-1
02/26/2020	-1
03/27/2020	-1
04/16/2020	-1
05/21/2020	-1
08/14/2020	-1
10/07/2020	-1
12/05/2020	-1
01/22/2021	0
02/03/2021	0
03/17/2021	0
04/23/2021	0
05/04/2021	0
06/30/2021	0
07/26/2021	0
12/27/2021	0
06/06/2022	0

date	previous_year
07/18/2022	0
11/14/2022	0
12/12/2022	0

Da -1 als Argument period_no in der Funktion inyear() verwendet wird, werden die Grenzen des Vergleichsjahrs um ein volles Jahr zurück verschoben. 2021 ist anfänglich als Vergleichsjahr identifiziert. Die period_no verschiebt das Vergleichsjahr um eins, sodass 2020 zum Vergleichsjahr wird.

Diagramm des Bereichs der Funktion inyear(), wobei das Argument „period_no“ auf -1 festgelegt ist.



Daher gibt jede Transaktion, die im Jahr 2020 stattfindet, ein boolesches Ergebnis von TRUE zurück.

Beispiel 3 – first_month_of_year

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Datensatz mit einer Reihe von Transaktionen zwischen 2020 und 2022, der in eine Tabelle namens „Transactions“ geladen wird
- Eine vorangehende load-Anweisung, die die Funktion inyear() enthält, die als das Feld „in_year“ festgelegt wird und bestimmt, welche Transaktionen im gleichen Jahr wie der 26. Juli 2021 stattfanden

In diesem Beispiel legt aber die Organisationsrichtlinie den März als ersten Monat des Geschäftsjahres fest.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';
Transactions:
    Load
```

```
        *,
        inyear(date,'07/26/2021', 0, 3) as in_year
        ;
Load
*
Inline
[
id,date,amount
8188,'01/13/2020',37.23
8189,'02/26/2020',17.17
8190,'03/27/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'08/14/2020',82.06
8194,'10/07/2020',40.39
8195,'12/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'12/27/2021',25.12
8204,'06/06/2022',46.23
8205,'07/18/2022',84.21
8206,'11/14/2022',96.24
8207,'12/12/2022',67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `in_year`

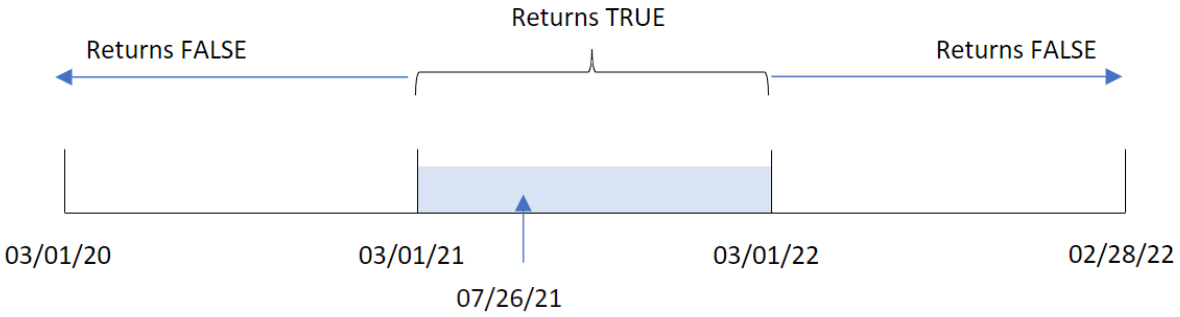
Ergebnistabelle

date	in_year
01/13/2020	0
02/26/2020	0
03/27/2020	0
04/16/2020	0
05/21/2020	0
08/14/2020	0
10/07/2020	0

date	in_year
12/05/2020	0
01/22/2021	0
02/03/2021	0
03/17/2021	-1
04/23/2021	-1
05/04/2021	-1
06/30/2021	-1
07/26/2021	-1
12/27/2021	-1
06/06/2022	0
07/18/2022	0
11/14/2022	0
12/12/2022	0

Da `3` als Argument `first_month_of_year` in der Funktion `inyear()` verwendet wird, beginnt das Jahr am 1. März und endet am Ende des Monats Februar.

Diagramm des Bereichs der Funktion `inyear()` mit März als erstem Monat des Jahres



Daher gibt jede Transaktion, die zwischen dem 1. März 2021 und dem 1. März 2022 stattfindet, ein boolesches Ergebnis von TRUE zurück.

Beispiel 4 – Diagrammobjektbeispiel

Ladeskript und Diagrammformel

Übersicht

Es werden derselbe Datensatz und dasselbe Szenario wie im ersten Beispiel verwendet.

In diesem Beispiel wird jedoch der unveränderte Datensatz in die Anwendung geladen. Die Berechnung, die bestimmt, ob Transaktionen im gleichen Jahr wie der 26. Juli 2021 stattfanden, wird als Kennzahl in einem Diagrammobjekt der Anwendung erstellt.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';
Transactions:
Load
*
Inline
[
id,date,amount
8188,'01/13/2020',37.23
8189,'02/26/2020',17.17
8190,'03/27/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'08/14/2020',82.06
8194,'10/07/2020',40.39
8195,'12/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'12/27/2021',25.12
8204,'06/06/2022',46.23
8205,'07/18/2022',84.21
8206,'11/14/2022',96.24
8207,'12/12/2022',67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu:

- `date`

Um zu berechnen, ob Transaktionen im selben Jahr wie der 26. Juli 2021 stattfanden, erstellen Sie die folgende Kennzahl:

- `=inyear(date,'07/26/2021', 0)`

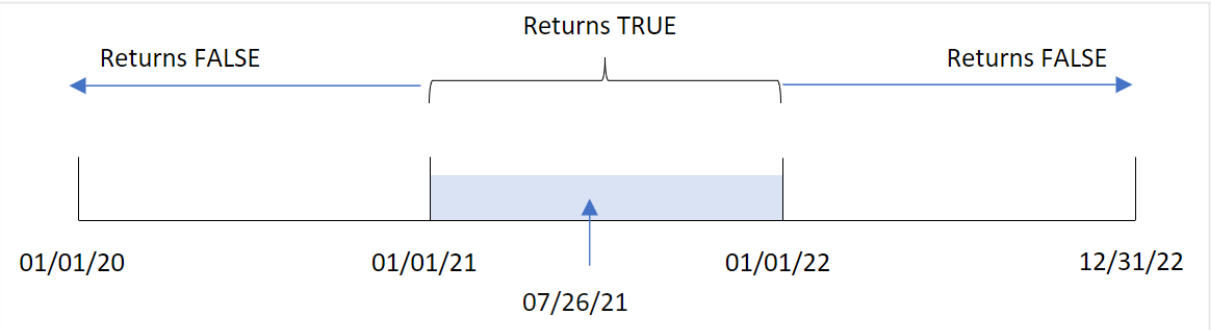
Ergebnistabelle

date	**=inyear(date,'07/26/2021',0)**
01/13/2020	0

date	=inyear(date,'07/26/2021',0)
02/26/2020	0
03/27/2020	0
04/16/2020	0
05/21/2020	0
08/14/2020	0
10/07/2020	0
12/05/2020	0
01/22/2021	-1
02/03/2021	-1
03/17/2021	-1
04/23/2021	-1
05/04/2021	-1
06/30/2021	-1
07/26/2021	-1
12/27/2021	-1
06/06/2022	0
07/18/2022	0
11/14/2022	0
12/12/2022	0

Das Feld „in_year“ wird im Diagramm mithilfe der Funktion `inyear()` erstellt. Das erste Argument identifiziert, welches Feld ausgewertet wird. Das zweite Argument ist ein hartcodiertes Datum, der 26. Juli 2021. Dies ist das `base_date`, das das Vergleichsjahr bestimmt. Eine `period_no` von 0 ist das abschließende Argument, was bedeutet, dass die Funktion `inyear()` keine Jahre vor oder nach dem Jahr vergleicht.

Diagramm des Bereichs der Funktion `inyear()` mit dem 27. Juli als Basisdatum.



Jede Transaktion, die im Jahr 2021 stattfindet, gibt ein boolesches Ergebnis von TRUE zurück.

Beispiel 5 – Szenario

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Datensatz, der in eine Tabelle namens „`Products`“ geladen wird
- Die Tabelle enthält die folgenden Felder:
    - product ID
    - product type
    - manufacture date
    - cost price

Der Endbenutzer möchte ein Diagrammobjekt, das nach Produkttyp die Kosten der im Jahr 2021 gefertigten Produkte anzeigt.

Ladeskript

```
Products:
Load
*
Inline
[
product_id,product_type,manufacture_date,cost_price
8188,product A,'01/13/2020',37.23
8189,product B,'02/26/2020',17.17
8190,product B,'03/27/2020',88.27
8191,product C,'04/16/2020',57.42
8192,product D,'05/21/2020',53.80
8193,product D,'08/14/2020',82.06
8194,product C,'10/07/2020',40.39
8195,product B,'12/05/2020',87.21
8196,product A,'01/22/2021',95.93
8197,product B,'02/03/2021',45.89
8198,product C,'03/17/2021',36.23
8199,product C,'04/23/2021',25.66
8200,product B,'05/04/2021',82.77
8201,product D,'06/30/2021',69.98
8202,product D,'07/26/2021',76.11
8203,product D,'12/27/2021',25.12
8204,product C,'06/06/2022',46.23
8205,product C,'07/18/2022',84.21
8206,product A,'11/14/2022',96.24
8207,product B,'12/12/2022',67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu:

- `product_type`

Um die Summe für jedes Produkt zu berechnen, das 2021 gefertigt wurde, erstellen Sie die folgende Kennzahl:

- `=sum(if(InYear(manufacture_date,makedate(2021,01,01),0),cost_price,0))`

Gehen Sie folgendermaßen vor:

1. Legen Sie das **Zahlenformat** der Kennzahl auf **Währung** fest.
2. Deaktivieren Sie unter **Darstellung** die Option **Gesamtwerte**.

Ergebnistabelle

product_type	=sum(if(InYear(manufacture_date,makedate(2021,01,01),0),cost_price,0))
Produkt A	$95.93
Produkt B	$128.66
Produkt C	$61.89
Produkt D	$171.21

Die Funktion `inyear()` gibt einen booleschen Wert zurück, wenn sie das Herstellungsdatum der einzelnen Produkte auswertet. Für alle 2021 hergestellten Produkte gibt die Funktion `inyear()` einen booleschen Wert von TRUE zurück und zeigt die Summe für `cost_price` an.

inyeartodate

Diese Funktion liefert True, wenn **timestamp** in dem laufenden Jahr mit **base_date** bis einschließlich der letzten Millisekunde von **base_date** liegt.

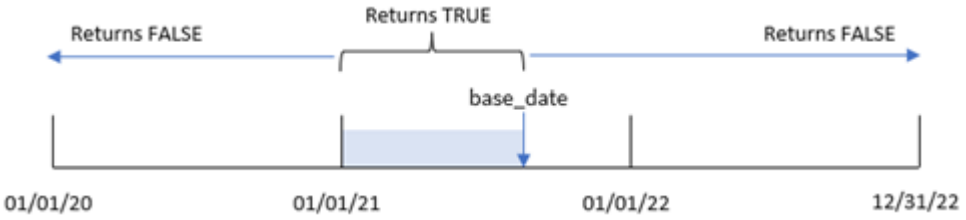
Syntax:

```
InYearToDate (timestamp, base_date, period_no[, first_month_of_year])
```

Rückgabe Datentyp: Boolesch

In Qlik Sense wird der boolesche Wert „wahr“ durch -1 dargestellt, der Wert „falsch“ durch 0.

Diagramm der Funktion „inyeartodate“



Die Funktion `inyeartodate()` segmentiert einen bestimmten Teil des Jahres, wobei das `base_date` das späteste zulässige Datum für das Jahressegment definiert. Die Funktion wertet dann aus, ob ein Datumsfeld oder Wert in dieses Segment fällt und gibt ein boolesches Ergebnis zurück.

Argumente

Argument	Beschreibung
timestamp	Das Datum, das mit **base_date** verglichen werden soll.
base_date	Datum, das für die Interpretation des Jahres verwendet wird.
period_no	Mit **period_no** können Sie den Beginn des Jahres festlegen.**period_no** ist eine ganze Zahl, wobei 0 für das Jahr steht, das **base_date** enthält. Negative Werte von **period_no** stehen für vorangehende, positive Werte für nachfolgende Jahre.
first_month_of_year	Wenn Sie mit abweichenden Geschäftsjahren arbeiten möchten, definieren Sie mit einer Zahl zwischen 2 und 12 für **first_month_of_year** einen anderen Startmonat.

Verwendung

Die Funktion `inyeartodate()` gibt einen booleschen Wert zurück. In der Regel wird dieser Funktionstyp als Bedingung in einem `if`-Ausdruck verwendet. Damit wird eine Aggregierung oder Berechnung zurückgegeben, abhängig davon, ob ein ausgewertetes Datum in das Jahr bis einschließlich zum betreffenden Datum fällt.

Beispielsweise kann die Funktion `inyeartodate()` verwendet werden, um alle in einem Jahr bis zu einem bestimmten Datum gefertigten Geräte zu identifizieren.

In diesen Beispielen wird das Datumsformat MM/TT/JJJJ verwendet. Das Datumsformat wird im Befehl `SET DateFormat` oben in Ihrem Datenladeskript angegeben. Ändern Sie das Format in den Beispielen nach Bedarf.

Funktionsbeispiele

Beispiel	Ergebnis
`inyeartodate ('01/25/2013', '02/01/2013', 0)`	Gibt `TRUE` zurück.
`inyeartodate ('01/25/2012', '01/01/2013', 0)`	Gibt `FALSE` zurück.

Beispiel	Ergebnis
`inyeartodate ('01/25/2012', '02/01/2013', -1)`	Gibt TRUE zurück.
`inyeartodate ('11/25/2012', '01/31/2013', 0, 4)`	Liefert TRUE. Der Wert von `timestamp` fällt in das Geschäftsjahr, das im vierten Monat beginnt, und vor den Wert von `base_date`.
`inyeartodate ('3/31/2013', '01/31/2013', 0, 4 )`	Liefert FALSE. Im Vergleich zum vorherigen Beispiel, liegt der Wert von `timestamp` noch im Geschäftsjahr, aber nach dem Wert von `base_date` und deshalb außerhalb des Teils des Jahres.

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Beispiel 1 – keine zusätzlichen Argumente

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz, der eine Reihe von Transaktionen zwischen 2020 und 2022 enthält, wird in eine Tabelle namens `Transactions` geladen.
- Das Datumsfeld wird im Format der Systemvariablen `DateFormat` (MM/TT/JJJJ) bereitgestellt.
- Die Erstellung eines Felds `in_year_to_date`, das bestimmt, welche Transaktionen im Jahr bis zum 26. Juli 2021 stattfanden.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        inyeartodate(date,'07/26/2021', 0) as in_year_to_date
        ;
Load
*
Inline
[
id,date,amount
8188,'01/13/2020',37.23
8189,'02/26/2020',17.17
8190,'03/27/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'06/14/2020',82.06
8194,'08/07/2020',40.39
8195,'09/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'07/27/2021',25.12
8204,'06/06/2022',46.23
8205,'07/18/2022',84.21
8206,'11/14/2022',96.24
8207,'12/12/2022',67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `in_year_to_date`

Ergebnistabelle

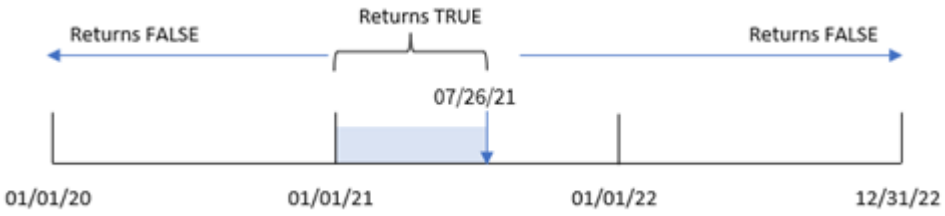
date	in_year_to_date
01/13/2020	0
02/26/2020	0
03/27/2020	0

date	in_year_to_date
04/16/2020	0
05/21/2020	0
06/14/2020	0
08/07/2020	0
09/05/2020	0
01/22/2021	-1
02/03/2021	-1
03/17/2021	-1
04/23/2021	-1
05/04/2021	-1
06/30/2021	-1
07/26/2021	-1
07/27/2021	0
06/06/2022	0
07/18/2022	0
11/14/2022	0
12/12/2022	0

Das Feld „in_year_to_date“ wird im vorangehenden load-Befehl mithilfe der Funktion `inyeartodate()` erstellt. Das erste angegebene Argument identifiziert, welches Feld ausgewertet wird.

Das zweite Argument ist ein hardcodiertes Datum für den 26. Juli 2021. Dies ist das `base_date`, das die Endbegrenzung des Jahressegments identifiziert. Eine `period_no` von 0 ist das abschließende Argument, was bedeutet, dass die Funktion keine Jahre vor oder nach dem segmentierten Jahr vergleicht.

Diagramm der Funktion „inyeartodate“, keine zusätzlichen Argumente



Jede Transaktion, die zwischen dem 1. Januar und 26. Juli stattfindet, gibt ein boolesches Ergebnis von TRUE zurück. Transaktionen vor 2021 und nach dem 26. Juli 2021 geben FALSE zurück.

Beispiel 2 – period_no

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Derselbe Datensatz und dasselbe Szenario wie im ersten Beispiel.
- Die Erstellung eines Felds `previous_year_to_date`, das bestimmt, welche Transaktionen ein ganzes Jahr vor dem Jahressegment stattfanden, das am 26. Juli 2021 endet.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        inyeartodate(date,'07/26/2021', -1) as previous_year_to_date
        ;
Load
*
Inline
[
id,date,amount
8188,'01/13/2020',37.23
8189,'02/26/2020',17.17
8190,'03/27/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'06/14/2020',82.06
8194,'08/07/2020',40.39
8195,'09/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'07/27/2021',25.12
8204,'06/06/2022',46.23
8205,'07/18/2022',84.21
8206,'11/14/2022',96.24
8207,'12/12/2022',67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

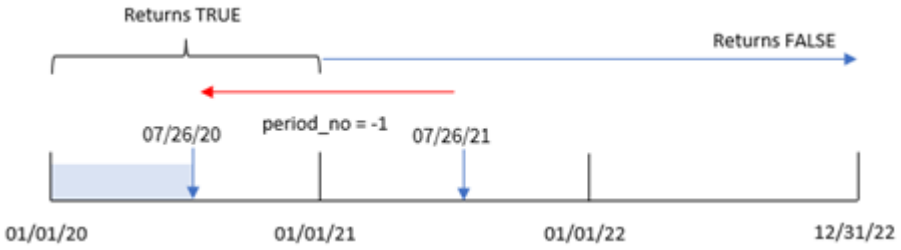
- `date`
- `previous_year_to_date`

Ergebnistabelle

date	previous_year_to_date
01/13/2020	-1
02/26/2020	-1
03/27/2020	-1
04/16/2020	-1
05/21/2020	-1
06/14/2020	-1
08/07/2020	0
09/05/2020	0
01/22/2021	0
02/03/2021	0
03/17/2021	0
04/23/2021	0
05/04/2021	0
06/30/2021	0
07/26/2021	0
07/27/2021	0
06/06/2022	0
07/18/2022	0
11/14/2022	0
12/12/2022	0

Ein `period_no`-Wert von -1 gibt an, dass die Funktion `inyeartodate ()` das Eingabequartalssegment mit dem vorherigen Jahr vergleicht. Mit dem Eingabedatum 26. Juli 2021 wurde das Segment vom 1. Januar 2021 bis 26. Juli 2021 zuerst als „Jahr bis dato“ identifiziert. Die `period_no` verschiebt dann dieses Segment um ein volles Jahr zurück, sodass die Datumsgrenzen der 1. Januar bis zum 26. Juli 2020 werden.

Diagramm der Funktion „inyeartodate“, Beispiel „period_no“



Daher gibt jede Transaktion, die zwischen dem 1. Januar und dem 26. Juli 2020 stattfindet, ein boolesches Ergebnis von TRUE zurück.

Beispiel 3 – first_month_of_year

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Derselbe Datensatz und dasselbe Szenario wie im ersten Beispiel.
- Die Erstellung eines Felds `in_year_to_date`, das bestimmt, welche Transaktionen im gleichen Jahr bis zum 26. Juli 2021 stattfanden.

In diesem Beispiel wird März als erster Monat des Geschäftsjahrs festgelegt.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        inyeartodate(date,'07/26/2021', 0,3) as in_year_to_date
        ;
Load
*
Inline
[
id,date,amount
8188,'01/13/2020',37.23
8189,'02/26/2020',17.17
8190,'03/27/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'06/14/2020',82.06
8194,'08/07/2020',40.39
8195,'09/05/2020',87.21
8196,'01/22/2021',95.93
```

```
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'07/27/2021',25.12
8204,'06/06/2022',46.23
8205,'07/18/2022',84.21
8206,'11/14/2022',96.24
8207,'12/12/2022',67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `in_year_to_date`

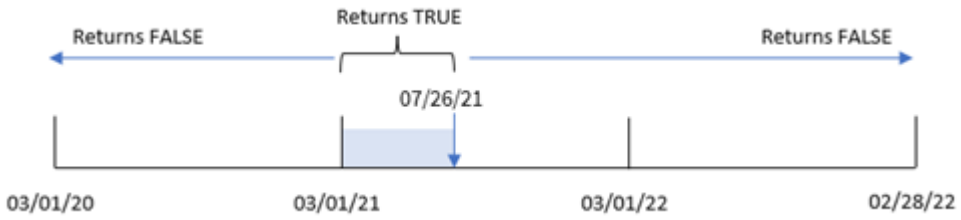
Ergebnistabelle

date	in_year_to_date
01/13/2020	0
02/26/2020	0
03/27/2020	0
04/16/2020	0
05/21/2020	0
06/14/2020	0
08/07/2020	0
09/05/2020	0
01/22/2021	0
02/03/2021	0
03/17/2021	-1
04/23/2021	-1
05/04/2021	-1
06/30/2021	-1
07/26/2021	-1
07/27/2021	0
06/06/2022	0

date	in_year_to_date
07/18/2022	0
11/14/2022	0
12/12/2022	0

Indem 3 als das Argument `first_month_of_year` in der Funktion `inyeartodate()` verwendet wird, beginnt die Funktion das Jahr am 1. März. Das `base_date` vom 26. Juli 2021 legt dann das Enddatum für dieses Jahressegment fest.

Diagramm der Funktion „inyeartodate“, Beispiel „first_month_of_year“



Daher gibt jede Transaktion, die zwischen dem 1. März und dem 26. Juli 2021 stattfand, ein boolesches Ergebnis von TRUE zurück, während Transaktionen mit Datum außerhalb dieser Grenzen einen Wert von FALSE zurückgeben.

Beispiel 4 – Diagrammobjektbeispiel

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript verwendet den gleichen Datensatz und das gleiche Szenario wie das erste Beispiel. In diesem Beispiel wird jedoch der unveränderte Datensatz in die Anwendung geladen. Die Berechnung, die bestimmt, ob Transaktionen im gleichen Jahr bis zum 26. Juli 2021 stattfanden, wird als Kennzahl in einem Diagrammobjekt der Anwendung erstellt.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8188,'01/13/2020',37.23
8189,'02/26/2020',17.17
```

```
8190,'03/27/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'06/14/2020',82.06
8194,'08/07/2020',40.39
8195,'09/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'07/27/2021',25.12
8204,'06/06/2022',46.23
8205,'07/18/2022',84.21
8206,'11/14/2022',96.24
8207,'12/12/2022',67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu:`date`.

Erstellen Sie die folgende Kennzahl:

```
=inyeartodate(date,'07/26/2021', 0)
```

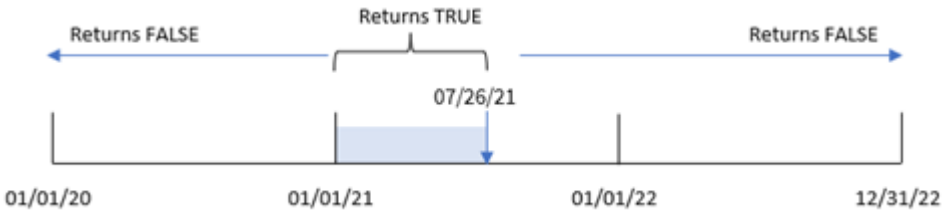
Ergebnistabelle

date	**=inyeartodate(date,'07/26/2021', 0)**
01/13/2020	0
02/26/2020	0
03/27/2020	0
04/16/2020	0
05/21/2020	0
06/14/2020	0
08/07/2020	0
09/05/2020	0
01/22/2021	-1
02/03/2021	-1
03/17/2021	-1
04/23/2021	-1

date	=inyeartodate(date,'07/26/2021', 0)
05/04/2021	-1
06/30/2021	-1
07/26/2021	-1
07/27/2021	0
06/06/2022	0
07/18/2022	0
11/14/2022	0
12/12/2022	0

Die Kennzahl `in_year_to_date` wird im Diagrammobjekt anhand der Funktion `inyeartodate()` erstellt. Das erste angegebene Argument identifiziert, welches Feld ausgewertet wird. Das zweite Argument ist ein hardcodiertes Datum für den 26. Juli 2021. Dies ist das `base_date`, das die Endbegrenzung des Vergleichsjahressegments identifiziert. Eine `period_no` von 0 ist das abschließende Argument, was bedeutet, dass die Funktion keine Jahre vor oder nach dem segmentierten Jahr vergleicht.

Diagramm der Funktion „inyeartodate“, Diagrammobjektbeispiel



Jede Transaktion, die zwischen dem 1. Januar und dem 26. Juli 2021 stattfindet, gibt ein boolesches Ergebnis von TRUE zurück. Transaktionen mit Datum vor 2021 und nach dem 26. Juli 2021 geben FALSE zurück.

Beispiel 5 – Szenario

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz, der in eine Tabelle namens `Products` geladen wird.
- Informationen bezüglich Produkt-ID, Produkttyp, Herstellungsdatum und Selbstkosten.

Der Endbenutzer möchte ein Diagrammobjekt, das nach Produkttyp die Kosten der im Jahr 2021 bis zum 26. Juli gefertigten Produkte anzeigt.

Ladeskript

```
Products:
Load
*
Inline
[
product_id,product_type,manufacture_date,cost_price
8188,product A,'01/13/2020',37.23
8189,product B,'02/26/2020',17.17
8190,product B,'03/27/2020',88.27
8191,product C,'04/16/2020',57.42
8192,product D,'05/21/2020',53.80
8193,product D,'08/14/2020',82.06
8194,product C,'10/07/2020',40.39
8195,product B,'12/05/2020',87.21
8196,product A,'01/22/2021',95.93
8197,product B,'02/03/2021',45.89
8198,product C,'03/17/2021',36.23
8199,product C,'04/23/2021',25.66
8200,product B,'05/04/2021',82.77
8201,product D,'06/30/2021',69.98
8202,product D,'07/26/2021',76.11
8203,product D,'12/27/2021',25.12
8204,product C,'06/06/2022',46.23
8205,product C,'07/18/2022',84.21
8206,product A,'11/14/2022',96.24
8207,product B,'12/12/2022',67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu:`product_type`.

Erstellen Sie eine Kennzahl, die die Summe für jedes Produkt berechnet, das 2021 vor dem 27. Juli gefertigt wurde:

```
=sum(if(inyeartodate(manufacture_date,makedate(2021,07,26),0),cost_price,0))
```

Legen Sie das **Zahlenformat** der Kennzahl auf **Währung** fest.

Ergebnistabelle

product_type	**=sum(if(inyeartodate(manufacture_date,makedate (2021,07,26),0),cost_price,0))**
Produkt A	$95.93
Produkt B	$128.66
Produkt C	$61.89
Produkt D	$146.09

Die Funktion `inyeartodate()` gibt einen booleschen Wert zurück, wenn sie das Herstellungsdatum der einzelnen Produkte auswertet. Für alle 2021 vor dem 27. Juli hergestellten Produkte gibt die Funktion `inyeartodate()` einen booleschen Wert von `TRUE` zurück und summiert `cost_price`.

Produkt D ist das einzige Produkt, das auch nach dem 26. Juli 2021 hergestellt wurde. Der Eintrag mit `product_ID` 8203 wurde am 27. Dezember gefertigt, und die Kosten betrugen $25.12. Daher wurden diese Kosten nicht in den Gesamtbetrag für Produkt D im Diagrammobjekt eingeschlossen.

lastworkdate

Die Funktion **lastworkdate** liefert das früheste Enddatum zur Vollendung von **no_of_workdays** (Montag bis Freitag) ab einem vorgegebenen **start_date** unter Berücksichtigung eventueller **holiday**.**start_date** und **holiday** müssen gültige Daten und Zeitstempel sein.

Syntax:

```
lastworkdate(start_date, no_of_workdays {, holiday})
```

Rückgabe Datentyp: ganze Zahl

Ein Kalender zeigt, wie die Funktion `lastworkdate()` verwendet wird

Sun	Mon	Tue	Wed	Thu	Fri	Sat
	1	2	3	4	5	6
7	8	9	10 start_date	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26 end_date	27
28	29	30	31			

Beschränkungen

Es gibt keine Methode zum Ändern der Funktion `lastworkdate()` für Regionen oder Szenarios für andere Elemente als eine Arbeitswoche, die am Montag beginnt und am Freitag endet.

Der Parameter „holiday“ muss eine String-Konstante sein. Er akzeptiert keine Formeln.

Verwendung

Die Funktion `lastworkdate()` wird häufig als Teil einer Formel verwendet, wenn der Benutzer das vorgeschlagene Enddatum eines Projekts oder einer Aufgabe berechnen und berücksichtigen möchte, wann das Projekt beginnt und welche Feiertage in dem Zeitraum liegen.

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Argumente

Argument	Beschreibung
start_date	Das auszuwertende Startdatum.
no_of_ workdays	Die Anzahl der zu erreichenden Arbeitstage.
holiday	Feiertagszeiträume, die von den Arbeitstagen auszuschließen sind. Ein Feiertag wird als ein Datum mit Zeichenfolgenkonstante angegeben. Sie können mehrere Feiertagstermine getrennt durch Kommas festlegen. **Beispiel:** `'12/25/2013', '12/26/2013', '12/31/2013', '01/01/2014'`

Beispiel 1 – einfaches Beispiel

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Datensatz mit Projekt-IDs, Projektstartdatumswerten und dem geschätzten Aufwand in Tagen, der für die Projekte erforderlich ist Der Datensatz wird in eine Tabelle namens „`Projects`“ geladen.

- Ein vorangehender load-Befehl, der die Funktion `lastworkdate()` enthält, die als das Feld „`end_date`“ festgelegt ist und identifiziert, wann das Ende für jedes Projekt geplant ist.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Projects:
    Load
       *,
       LastWorkDate(start_date,effort) as end_date
       ;
Load
id,
start_date,
effort
Inline
[
id,start_date,effort
1,01/01/2022,14
2,02/10/2022,17
3,05/17/2022,5
4,06/01/2022,12
5,08/10/2022,26
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `id`
- `start_date`
- `effort`
- `end_date`

Ergebnistabelle

id	start_date	effort	end_date
1	01/01/2022	14	01/20/2022
2	02/10/2022	17	03/04/2022
3	05/17/2022	5	05/23/2022
4	06/01/2022	12	06/16/2022
5	08/10/2022	26	09/14/2022

Da keine geplanten Feiertage vorhanden sind, fügt die Funktion die definierte Anzahl Werktage (Montag bis Freitag) zum Startdatum hinzu, um das früheste mögliche Enddatum zu finden.

Der folgende Kalender zeigt das Start- und Enddatum für Projekt 3, wobei die Werktage grün hervorgehoben sind.

Ein Kalender zeigt das Start- und Enddatum von Projekt 3

Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17 Start Date	18	19	20	21
22	23 End Date	24	25	26	27	28
29	30	31				

Beispiel 2 – Ein Feiertag

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Datensatz mit Projekt-IDs, Projektstartdatumswerten und dem geschätzten Aufwand in Tagen, der für die Projekte erforderlich ist Der Datensatz wird in eine Tabelle namens „`Projects`“ geladen.
- Eine vorangehende load-Anweisung, die die Funktion `lastworkdate()` enthält, das als das Feld „`end_date`“ festgelegt ist und identifiziert, wann das Ende für jedes Projekt geplant ist

Es ist jedoch ein Feiertag am 18. Mai 2022 geplant. Die Funktion `lastworkdate()` in der vorangehenden load-Anweisung schließt den Feiertag in ihr drittes Argument ein, um zu identifizieren, wann das Ende für jedes Projekt geplant ist.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Projects:
    Load
       *,
       LastWorkDate(start_date,effort, '05/18/2022') as end_date
       ;
Load
id,
start_date,
effort
Inline
[
id,start_date,effort
1,01/01/2022,14
2,02/10/2022,17
3,05/17/2022,5
4,06/01/2022,12
5,08/10/2022,26
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `id`
- `start_date`
- `effort`
- `end_date`

Ergebnistabelle

id	start_date	effort	end_date
1	01/01/2022	14	01/20/2022
2	02/10/2022	17	03/04/2022
3	05/17/2022	5	05/24/2022
4	06/01/2022	12	06/16/2022
5	08/10/2022	26	09/14/2022

Der eintägige geplante Feiertag wird als drittes Argument in die Funktion `lastworkdate()` eingegeben. Als Ergebnis wird das Enddatum für Projekt 3 auf einen Tag später verschoben, weil der Feiertag auf einem der Werktage vor dem Enddatum liegt.

Der folgende Kalender zeigt das Start- und Enddatum für Projekt 3 und zeigt, dass der Feiertag das Enddatum des Projekts um einen Tag verändert.

Ein Kalender zeigt das Start- und Enddatum von Projekt 3 mit einem Feiertag am 18. Mai

Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17 Start Date	18 Holiday	19	20	21
22	23	24 End Date	25	26	27	28
29	30	31				

Beispiel 3 – Mehrere Feiertage

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Datensatz mit Projekt-IDs, Projektstartdatumswerten und dem geschätzten Aufwand in Tagen, der für die Projekte erforderlich ist Der Datensatz wird in eine Tabelle namens „`Projects`“ geladen.

- Eine vorangehende load-Anweisung, die die Funktion `lastworkdate()` enthält, die als das Feld „`end_date`“ festgelegt ist und identifiziert, wann das Ende für jedes Projekt geplant ist

Zwischen dem 19., 20., 21. und 22 Mai sind jedoch Feiertage geplant. Die Funktion `lastworkdate()` in der vorangehenden load-Anweisung schließt jeden der Feiertage in ihr drittes Argument ein, um zu identifizieren, wann das Ende für jedes Projekt geplant ist.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Projects:
    Load
       *,
       LastWorkDate(start_date,effort, '05/19/2022','05/20/2022','05/21/2022','05/22/2022') as
end_date
       ;
Load
id,
start_date,
effort
Inline
[
id,start_date,effort
1,01/01/2022,14
2,02/10/2022,17
3,05/17/2022,5
4,06/01/2022,12
5,08/10/2022,26
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `id`
- `start_date`
- `effort`
- `end_date`

Ergebnistabelle

id	start_date	effort	end_date
1	01/01/2022	14	01/20/2022
2	02/10/2022	17	03/04/2022
3	05/17/2022	5	05/25/2022
4	06/01/2022	12	06/16/2022
5	08/10/2022	26	09/14/2022

Die vier Feiertage werden als Liste von Argumenten nach dem Startdatum und der Anzahl der Werktage in die Funktion `lastworkdate()` eingegeben.

Der folgende Kalender zeigt das Start- und Enddatum für Projekt 3 und zeigt, dass die Feiertage das Enddatum des Projekts um drei Tage verändern.

Ein Kalender zeigt das Start- und Enddatum von Projekt 3 mit Feiertagen vom 19. bis 22. Mai

Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17 Start Date	18	19 Holiday	20 Holiday	21 Holiday
22 Holiday	23	24	25 End Date	26	27	28
29	30	31				

Beispiel 4 – Ein Feiertag (Diagramm)

Ladeskript und Diagrammformel

Übersicht

Es werden derselbe Datensatz und dasselbe Szenario wie im ersten Beispiel verwendet.

In diesem Beispiel wird jedoch der unveränderte Datensatz in die App geladen. Das Feld `end_date` wird als Kennzahl in einem Diagrammobjekt berechnet.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';
```

```
Projects:
Load
id,
start_date,
effort
Inline
[
id,start_date,effort
1,01/01/2022,14
2,02/10/2022,17
3,05/17/2022,5
4,06/01/2022,12
5,08/10/2022,26
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `id`
- `start_date`
- `effort`

Erstellen Sie die folgende Kennzahl, um „end_date“ zu berechnen:

- `=LastWorkDate(start_date,effort,'05/18/2022')`

Ergebnistabelle

id	start_date	effort	=LastWorkDate(start_date,effort,'05/18/2022')
1	01/01/2022	14	01/20/2022
2	02/10/2022	17	03/04/2022
3	05/17/2022	5	05/23/2022
4	06/01/2022	12	06/16/2022
5	08/10/2022	26	09/14/2022

Der einzige geplante Feiertag wird als Kennzahl in das Diagramm eingegeben. Als Ergebnis wird das Enddatum für Projekt 3 auf einen Tag später verschoben, weil der Feiertag auf einem der Werktage vor dem Enddatum liegt.

Der folgende Kalender zeigt das Start- und Enddatum für Projekt 3 und zeigt, dass der Feiertag das Enddatum des Projekts um einen Tag verändert.

Ein Kalender zeigt das Start- und Enddatum von Projekt 3 mit einem Feiertag am 18. Mai

Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17 Start Date	18 Holiday	19	20	21
22	23	24 End Date	25	26	27	28
29	30	31				

localtime

Diese Funktion liefert einen Zeitstempel der aktuellen Uhrzeit, bezogen auf eine bestimmte Zeitzone.

Syntax:

```
LocalTime([timezone [, ignoreDST ]])
```

Rückgabe Datentyp: dual

Argumente

Argument	Beschreibung
timezone	Für den Parameter **timezone** ist ein String anzugeben, der einer Ortsbezeichnung entspricht, wie sie in der **Windows-Systemsteuerung** bei **Zeitzone** unter **Datum und Uhrzeit** angegeben ist. Auch Angaben der Form 'GMT+hh:mm' können verwendet werden. Eine Liste mit akzeptierten Orten und Zeitzonen ist auch in der unten stehenden Tabelle aufgeführt. Wenn keine Zeitzone angegeben ist, wird die lokale Uhrzeit ausgegeben. *Falls Sie eine DST-Verschiebung verwenden (Sie also einen Argumentwert **ignoreDST** angeben, der den Wert False setzt), müssen Sie einen Ort anstatt einer GMT-Verschiebung im **place**-Argument angeben. Dies beruht darauf, dass für die Anpassung der Sommerzeit zusätzlich zu den Längengradinformationen, die durch eine GMT-Verschiebung bereitgestellt werden, Breitengradinformationen erforderlich sind. Weitere Informationen finden Sie unter Verwenden von GMT-Verschiebungen in Verbindung mit DST (page 871).*
ignoreDST	Falls dieses Argument den Wert True setzt, wird DST (Sommerzeit) ignoriert. Gültige Argumentwerte, die auf True gesetzt werden, enthalten `-1` und `True()`. Falls dieses Argument den Wert auf False setzt, wird der Zeitstempel für die Sommerzeit angepasst. Gültige Argumentwerte, die auf False gesetzt werden, enthalten `0` und `False()`. Falls das Argument **ignoreDST** ungültig ist, wertet die Funktion den Ausdruck aus, als ob der Wert **ignore_dst** den Wert True festlegt. Falls das Argument **ignoreDST** nicht angegeben wird, wertet die Funktion den Ausdruck aus, als ob der Wert **ignore_dst** den Wert False festlegt.

Gültige Orte und Zeitzonen

A-C	D-K	L-R	S-Z
Abu Dhabi	Darwin	La Paz	Samoa
Adelaide	Dhaka	Lima	Santiago
Alaska	Eastern Time (US & Canada)	Lisbon	Sapporo
Amsterdam	Edinburgh	Ljubljana	Sarajevo
Arizona	Ekaterinburg	London	Saskatchewan
Astana	Fiji	Madrid	Seoul
Athens	Georgetown	Magadan	Singapore

A-C	D-K	L-R	S-Z
Atlantic Time (Canada)	Greenland	Mazatlan	Skopje
Auckland	Greenwich Mean Time : Dublin	Melbourne	Sofia
Azores	Guadalajara	Mexico City	Solomon Is.
Baghdad	Guam	Mid-Atlantic	Sri Jayawardenepura
Baku	Hanoi	Minsk	St. Petersburg
Bangkok	Harare	Monrovia	Stockholm
Beijing	Hawaii	Monterrey	Sydney
Belgrade	Helsinki	Moscow	Taipei
Berlin	Hobart	Mountain Time (US & Canada)	Tallinn
Bern	Hong Kong	Mumbai	Tashkent
Bogota	Indiana (East)	Muscat	Tbilisi
Brasilia	International Date Line West	Nairobi	Tehran
Bratislava	Irkutsk	New Caledonia	Tokyo
Brisbane	Islamabad	New Delhi	Urumqi
Brussels	Istanbul	Newfoundland	Warsaw
Bucharest	Jakarta	Novosibirsk	Wellington
Budapest	Jerusalem	Nuku'alofa	West Central Africa
Buenos Aires	Kabul	Osaka	Vienna
Cairo	Kamchatka	Pacific Time (US & Canada)	Vilnius
Canberra	Karachi	Paris	Vladivostok
Cape Verde Is.	Kathmandu	Perth	Volgograd
Caracas	Kolkata	Port Moresby	Yakutsk
Casablanca	Krasnoyarsk	Prague	Yerevan
Central America	Kuala Lumpur	Pretoria	Zagreb
Central Time (US & Canada)	Kuwait	Quito	-
Chennai	Kyiv	Riga	-

A-C	D-K	L-R	S-Z
Chihuahua	-	Riyadh	-
Chongqing	-	Rome	-
Copenhagen	-	-	-

Beispiele und Ergebnisse:

Die unten stehenden Beispiele basieren auf der Funktion, die mit dem Zeitstempel 2023-08-14 08:39:47 lokaler Zeit aufgerufen wurde, wobei die lokale Zeitzone der Server- oder Desktopumgebung GMT-05:00 ist und sich in einer Region befindet, die seit diesem aufgelisteten Datum die Sommerzeit eingeführt hat.

Skriptbeispiele

Beispiel	Ergebnis
`localtime ()`	Liefert die lokale Zeit 2023-08-14 08:39:47.
`localtime ('London')`	Liefert die lokale Zeit 2023-08-14 13:39:47 in London.
`localtime ('GMT+02:00')`	Liefert die lokale Zeit in der Zeitzone GMT+02:00, 2023-08-14 14:39:47. Es wird keine Anpassung für die Sommerzeit vorgenommen, da eine GMT-Verschiebung und kein Ort angegeben ist.
`localtime ('Paris',-1)`	Liefert die lokale Zeit in Paris ohne Berücksichtigung der Sommerzeit, 2023-08-14 13:39:47.
`localtime ('Paris',True())`	Liefert die lokale Zeit in Paris ohne Berücksichtigung der Sommerzeit, 2023-08-14 13:39:47.
`localtime ('Paris',0)`	Liefert die lokale Zeit in Paris unter Berücksichtigung der Sommerzeit, 2023-08-14 14:39:47.
`localtime ('Paris',False ())`	Liefert die lokale Zeit in Paris unter Berücksichtigung der Sommerzeit, 2023-08-14 14:39:47.

Verwenden von GMT-Verschiebungen in Verbindung mit DST

Anhand der Implementierung der ICU-Bibliotheken (International Components for Unicode in Qlik Sense erfordert die Nutzung von GMT-Verschiebungen (Greenwich Mean Time) in Kombination mit DST (Sommerzeit) zusätzliche Breitengradinformationen.

GMT ist eine Breitengradverschiebung (Ost-West), wohingegen DST eine Längengradverschiebung (Nord-Süd) ist. Helsinki (Finnland) und Johannesburg (Südafrika) haben dieselbe Verschiebung von GMT+02:00, allerdings nicht dieselbe DST-Verschiebung. Dies bedeutet, dass eine beliebige DST-Verschiebung außer der GMT-Verschiebung Informationen zur Breitengradposition der lokalen Zeitzone (geografische Zeitzoneneingabe) benötigt, damit vollständige Informationen über lokale DST-Bedingungen vorliegen.

lunarweekend

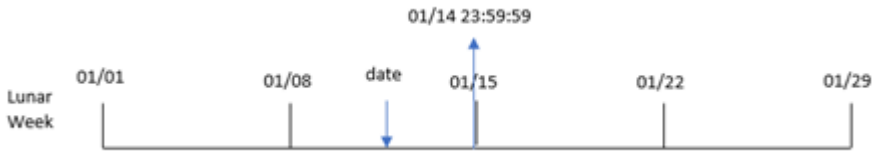
Diese Funktion liefert einen Wert, der dem Zeitstempel der letzten Millisekunde des letzten Tags der Mondwoche entspricht, in der **date** liegt. Bei Mondwochen in Qlik Sense wird der 1. Januar als der erste Tag der Woche gezählt. Mit Ausnahme der letzten Woche des Jahres umfasst jede Woche genau sieben Tage.

Syntax:

```
LunarweekEnd(date[, period_no[, first_week_day]])
```

Rückgabe Datentyp: dual

Beispieldiagramm der Funktion lunarweekend()



Die Funktion `lunarweekend()` bestimmt, in welche Mondwoche `date` fällt. Sie gibt dann einen Zeitstempel im Datumsformat für die letzte Millisekunde dieser Woche zurück.

Argumente

Argument	Beschreibung
date	Datum oder Zeitstempel für die Evaluierung.
period_no	**period_no** ist eine ganze Zahl oder eine Formel, die eine ganze Zahl ergibt, wobei 0 für die Mondwoche steht, die **date** enthält. Negative Werte von **period_no** stehen für vorangehende, positive Werte für nachfolgende Mondwochen.
first_week_day	Ein Startwert, der größer oder kleiner als Null sein kann. Dadurch wird der Beginn des Jahres um die angegebene Anzahl an Tagen und/oder den Bruchteil eines Tages verschoben.

Verwendung

Die Funktion `lunarweekend()` wird in der Regel als Teil einer Formel verwendet, wenn in der Berechnung der Teil der Woche verwendet werden soll, der noch nicht verstrichen ist. Anders als bei der Funktion `weekend()` endet die letzte Mondwoche in jedem Kalenderjahr am 31. Dezember. Die Funktion `lunarweekend()` kann beispielsweise verwendet werden, um Zinsen zu berechnen, die während der Woche noch nicht angefallen sind.

Funktionsbeispiele

Beispiel	Ergebnis
`lunarweekend('01/12/2013')`	Gibt 01/14/2013 23:59:59 zurück.
`lunarweekend('01/12/2013', -1)`	Gibt 01/07/2013 23:59:59 zurück.
`lunarweekend('01/12/2013', 0, 1)`	Gibt 01/15/2013 23:59:59 zurück.

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Beispiel 1 – keine zusätzlichen Argumente

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz enthält eine Reihe von Transaktionen für 2022 und wird in eine Tabelle namens `Transactions` geladen.
- Das Datumsfeld wird im Format der Systemvariablen `DateFormat` (MM/TT/JJJJ) bereitgestellt.
- Erstellung eines Felds `end_of_week`, das den Zeitstempel für das Ende der Mondwoche zurückgibt, in der die Transaktion stattfand

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        lunarweekend(date) as end_of_week,
        timestamp(lunarweekend(date)) as end_of_week_timestamp
        ;
```

```
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `end_of_week`
- `end_of_week_timestamp`

Ergebnistabelle

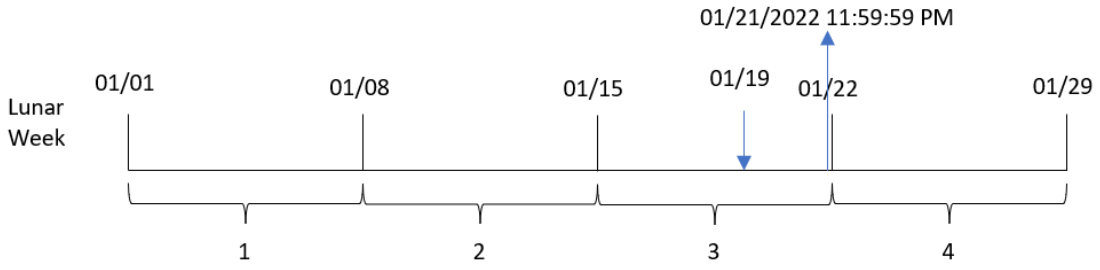
date	end_of_week	end_of_week_timestamp
1/7/2022	01/07/2022	1/7/2022 11:59:59 PM
1/19/2022	01/21/2022	1/21/2022 11:59:59 PM
2/5/2022	02/11/2022	2/11/2022 11:59:59 PM
2/28/2022	03/04/2022	3/4/2022 11:59:59 PM
3/16/2022	03/18/2022	3/18/2022 11:59:59 PM
4/1/2022	04/01/2022	4/1/2022 11:59:59 PM
5/7/2022	05/13/2022	5/13/2022 11:59:59 PM
5/16/2022	05/20/2022	5/20/2022 11:59:59 PM

date	end_of_week	end_of_week_timestamp
6/15/2022	06/17/2022	6/17/2022 11:59:59 PM
6/26/2022	07/01/2022	7/1/2022 11:59:59 PM
7/9/2022	07/15/2022	7/15/2022 11:59:59 PM
7/22/2022	07/22/2022	7/22/2022 11:59:59 PM
7/23/2022	07/29/2022	7/29/2022 11:59:59 PM
7/27/2022	07/29/2022	7/29/2022 11:59:59 PM
8/2/2022	08/05/2022	8/5/2022 11:59:59 PM
8/8/2022	08/12/2022	8/12/2022 11:59:59 PM
8/19/2022	08/19/2022	8/19/2022 11:59:59 PM
9/26/2022	09/30/2022	9/30/2022 11:59:59 PM
10/14/2022	10/14/2022	10/14/2022 11:59:59 PM
10/29/2022	11/04/2022	11/4/2022 11:59:59 PM

Das Feld `end_of_week` wird in dem vorangehenden load-Befehl erstellt, indem die Funktion `lunarweekend()` verwendet und das Feld `date` als Argument der Funktion übergeben wird.

Die Funktion `lunarweekend()` identifiziert, in welche Mondwoche der Datumswert fällt, und gibt einen Zeitstempel für die letzte Millisekunde dieser Woche zurück.

Diagramm der Funktion `lunarweekend()`, Beispiel ohne zusätzliche Argumente



Transaktion 8189 fand am 19. Januar statt. Die Funktion `lunarweekend()` identifiziert, dass die Mondwoche am 15. Januar beginnt. Daher gibt der Wert für `end_of_week` dieser Transaktion die letzte Millisekunde der Mondwoche zurück, also den 21. Januar um 11:59:59 PM.

Beispiel 2 – period_no

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Derselbe Datensatz und dasselbe Szenario wie im ersten Beispiel.
- Es wird ein Feld `previous_lunar_week_end` erstellt, das den Zeitstempel für das Ende der vor dem Transaktionsdatum liegenden Mondwoche zurückgibt.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        lunarweekend(date,-1) as previous_lunar_week_end,
        timestamp(lunarweekend(date,-1)) as previous_lunar_week_end_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

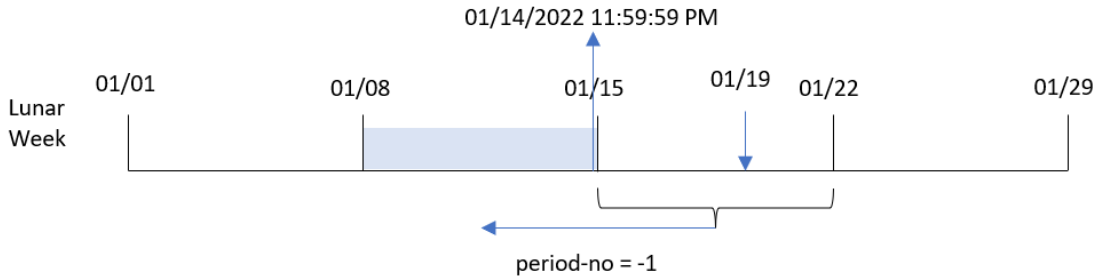
- `date`
- `previous_lunar_week_end`
- `previous_lunar_week_end_timestamp`

Ergebnistabelle

date	previous_lunar_week_end	previous_lunar_week_end_timestamp
1/7/2022	12/31/2021	12/31/2021 11:59:59 PM
1/19/2022	01/14/2022	1/14/2022 11:59:59 PM
2/5/2022	02/04/2022	2/4/2022 11:59:59 PM
2/28/2022	02/25/2022	2/25/2022 11:59:59 PM
3/16/2022	03/11/2022	3/18/2022 11:59:59 PM
4/1/2022	03/25/2022	3/25/2022 11:59:59 PM
5/7/2022	05/06/2022	5/6/2022 11:59:59 PM
5/16/2022	05/13/2022	5/13/2022 11:59:59 PM
6/15/2022	06/10/2022	6/10/2022 11:59:59 PM
6/26/2022	06/24/2022	6/24/2022 11:59:59 PM
7/9/2022	07/08/2022	7/8/2022 11:59:59 PM
7/22/2022	07/15/2022	7/15/2022 11:59:59 PM
7/23/2022	07/22/2022	7/22/2022 11:59:59 PM
7/27/2022	07/22/2022	7/22/2022 11:59:59 PM
8/2/2022	07/29/2022	7/29/2022 11:59:59 PM
8/8/2022	08/05/2022	8/5/2022 11:59:59 PM
8/19/2022	08/12/2022	8/12/2022 11:59:59 PM
9/26/2022	09/23/2022	9/23/2022 11:59:59 PM
10/14/2022	10/07/2022	10/7/2022 11:59:59 PM
10/29/2022	10/28/2022	10/28/2022 11:59:59 PM

Da in diesem Fall eine `period_no` von -1 als Versatzargument in der Funktion `lunarweekend()` verwendet wurde, identifiziert die Funktion zuerst die Mondwoche, in der die Transaktionen stattfanden. Dann geht sie eine Woche zurück und identifiziert die letzte Millisekunde dieser Mondwoche.

Diagramm der Funktion lunarweekend(), Beispiel „period_no“



Transaktion 8189 fand am 19. Januar statt. Die Funktion `lunarweekend()` identifiziert, dass die Mondwoche am 15. Januar beginnt. Daher begann die vorherige Mondwoche am 8. Januar und endete am 14. Januar um 11:59:59 PM; dies ist der Wert, der für das Feld `previous_lunar_week_end` zurückgegeben wird.

Beispiel 3 – first_week_day

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript verwendet den gleichen Datensatz und das gleiche Szenario wie das erste Beispiel. In diesem Beispiel wird für den Beginn der Mondwochen der 5. Januar festgelegt.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        lunarweekend(date,0,4) as end_of_week,
timestamp(lunarweekend(date,0,4)) as end_of_week_timestamp
;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
```

```
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `end_of_week`
- `end_of_week_timestamp`

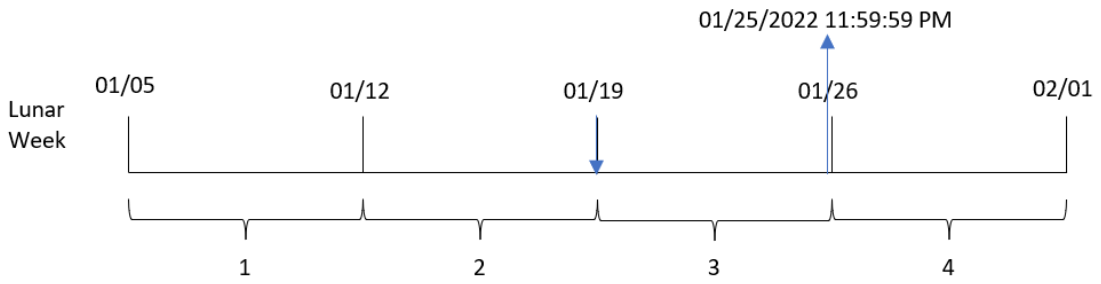
Ergebnistabelle

date	end_of_week	end_of_week_timestamp
1/7/2022	01/11/2022	1/11/2022 11:59:59 PM
1/19/2022	01/25/2022	1/25/2022 11:59:59 PM
2/5/2022	02/08/2022	2/8/2022 11:59:59 PM
2/28/2022	03/01/2022	3/1/2022 11:59:59 PM
3/16/2022	03/22/2022	3/22/2022 11:59:59 PM
4/1/2022	04/05/2022	4/5/2022 11:59:59 PM
5/7/2022	05/10/2022	5/10/2022 11:59:59 PM
5/16/2022	05/17/2022	5/17/2022 11:59:59 PM
6/15/2022	06/21/2022	6/21/2022 11:59:59 PM
6/26/2022	06/28/2022	6/28/2022 11:59:59 PM
7/9/2022	07/12/2022	7/12/2022 11:59:59 PM
7/22/2022	07/26/2022	7/26/2022 11:59:59 PM
7/23/2022	07/26/2022	7/26/2022 11:59:59 PM
7/27/2022	08/02/2022	8/2/2022 11:59:59 PM
8/2/2022	08/02/2022	8/2/2022 11:59:59 PM
8/8/2022	08/09/2022	8/9/2022 11:59:59 PM

date	end_of_week	end_of_week_timestamp
8/19/2022	08/23/2022	8/23/2022 11:59:59 PM
9/26/2022	09/27/2022	9/27/2022 11:59:59 PM
10/14/2022	10/18/2022	10/18/2022 11:59:59 PM
10/29/2022	11/01/2022	11/1/2022 11:59:59 PM

Da in diesem Fall das Argument `first_week_date` von 4 in der Funktion `lunarweekend()` verwendet wird, wird der Start des Jahres vom 1. auf den 5. Januar verschoben.

Diagramm der Funktion `lunarweekend()`, Beispiel „first_week_day“



Transaktion 8189 fand am 19. Januar statt. Da Mondwochen am 5. Januar beginnen, identifiziert die Funktion `lunarweekend()`, dass die Mondwoche, die den 19. Januar enthält, auch am 19. Januar beginnt. Daher liegt das Ende dieser Mondwoche am 25. Januar um 11:59:59 PM; dies ist der Wert, der für das Feld `end_of_week` zurückgegeben wird.

Beispiel 4 – Diagrammobjektbeispiel

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript verwendet den gleichen Datensatz und das gleiche Szenario wie das erste Beispiel.

In diesem Beispiel wird jedoch der unveränderte Datensatz in die Anwendung geladen. Die Berechnung, die einen Zeitstempel für das Ende der Mondwoche zurückgibt, in der die Transaktionen stattfanden, wird als Kennzahl in einem Diagrammobjekt der Anwendung erstellt.

Ladeskript

```
Transactions:
Load
*
Inline
[
```

```
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu: `date`.

Fügen Sie die folgenden Kennzahlen hinzu:

```
=lunarweekend(date)
```

```
=timestamp(lunarweekend(date))
```

Ergebnistabelle

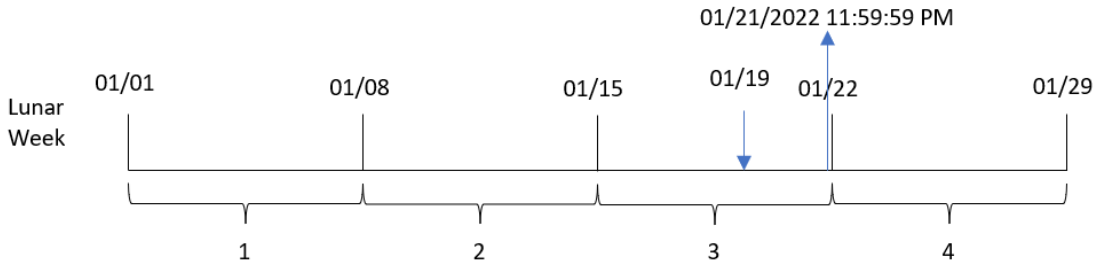
date	=lunarweekend(date)	=timestamp(lunarweekend(date))
1/7/2022	01/07/2022	1/7/2022 11:59:59 PM
1/19/2022	01/21/2022	1/21/2022 11:59:59 PM
2/5/2022	02/11/2022	2/11/2022 11:59:59 PM
2/28/2022	03/04/2022	3/4/2022 11:59:59 PM
3/16/2022	03/18/2022	3/18/2022 11:59:59 PM
4/1/2022	04/01/2022	4/1/2022 11:59:59 PM
5/7/2022	05/13/2022	5/13/2022 11:59:59 PM
5/16/2022	05/20/2022	5/20/2022 11:59:59 PM
6/15/2022	06/17/2022	6/17/2022 11:59:59 PM

date	=lunarweekend(date)	=timestamp(lunarweekend(date))
6/26/2022	07/01/2022	7/1/2022 11:59:59 PM
7/9/2022	07/15/2022	7/15/2022 11:59:59 PM
7/22/2022	07/22/2022	7/22/2022 11:59:59 PM
7/23/2022	07/29/2022	7/29/2022 11:59:59 PM
7/27/2022	07/29/2022	7/29/2022 11:59:59 PM
8/2/2022	08/05/2022	8/5/2022 11:59:59 PM
8/8/2022	08/12/2022	8/12/2022 11:59:59 PM
8/19/2022	08/19/2022	8/19/2022 11:59:59 PM
9/26/2022	09/30/2022	9/30/2022 11:59:59 PM
10/14/2022	10/14/2022	10/14/2022 11:59:59 PM
10/29/2022	11/04/2022	11/4/2022 11:59:59 PM

Die Kennzahl `end_of_week` wird im Diagrammobjekt erstellt, indem die Funktion `lunarweekend()` verwendet und das Feld `date` als Argument der Funktion übergeben wird.

Die Funktion `lunarweekend()` identifiziert, in welche Mondwoche der Datumswert fällt, und gibt einen Zeitstempel für die letzte Millisekunde dieser Woche zurück.

Diagramm der Funktion `lunarweekend()`, Diagrammobjektbeispiel



Transaktion 8189 fand am 19. Januar statt. Die Funktion `lunarweekend()` identifiziert, dass die Mondwoche am 15. Januar beginnt. Daher gibt der Wert für `end_of_week` dieser Transaktion die letzte Millisekunde der Mondwoche zurück, also den 21. Januar um 11:59:59 PM.

Beispiel 5 – Szenario

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Datensatz, der in eine Tabelle namens `Employee_Expenses` geladen wird
- Mitarbeiter-IDs, Mitarbeitername und die durchschnittlichen täglichen Spesenanträge jedes Mitarbeiters

Der Endbenutzer möchte ein Diagrammobjekt, das nach Mitarbeiter-ID und Mitarbeitername die geschätzten Spesenanträge anzeigt, die für die restliche Mondwoche noch anfallen.

Ladeskript

```
Employee_Expenses:
Load
*
Inline
[
employee_id,employee_name,avg_daily_claim
182,Mark, $15
183,Deryck, $12.5
184,Dexter, $12.5
185,Sydney,$27
186,Agatha,$18
];
```

Ergebnisse

Gehen Sie folgendermaßen vor:

1. Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle.
2. Fügen Sie die folgenden Felder als Dimensionen hinzu:
    - `employee_id`
    - `employee_name`
3. Erstellen Sie dann die folgende Kennzahl, um die kumulierten Zinsen zu berechnen: `=(lunarweekend(today(1))-today(1))*avg_daily_claim`
4. Legen Sie das **Zahlenformat** der Kennzahl auf **Währung** fest.

Ergebnistabelle

employee_id	employee_name	=(lunarweekend(today(1))-today(1))*avg_daily_claim
182	Mark	$75.00
183	Deryck	$62.50
184	Dexter	$62.50
185	Sydney	$135.00
186	Agatha	$90.00

Da das aktuelle Datum als einziges Argument verwendet wird, gibt die Funktion `lunarkweekend()` das Enddatum der aktuellen Mondwoche zurück. Dann zieht die Formel das aktuelle Datum vom Enddatum der Mondwoche ab und gibt die Anzahl der in dieser Woche verbleibenden Tage zurück.

Dieser Wert wird dann mit den durchschnittlichen täglichen Spesenanträgen der einzelnen Mitarbeitern multipliziert, um den geschätzten Spesenbetrag pro Mitarbeiter für den verbleibenden Teil der Mondwoche zu berechnen.

lunarweekname

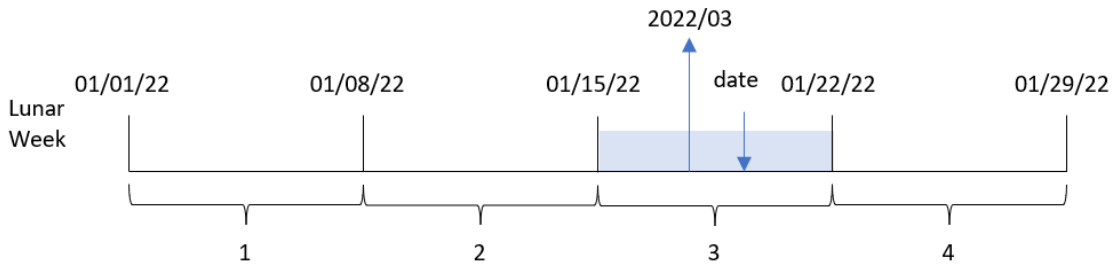
Diese Funktion liefert den Anzeigewert der ersten Millisekunde des ersten Tags der Mondwoche, in der **date** liegt. Das Ergebnis wird als Kombination von Jahr und Mondwochennummer formatiert. Bei Mondwochen in Qlik Sense wird der 1. Januar als der erste Tag der Woche gezählt. Mit Ausnahme der letzten Woche des Jahres umfasst jede Woche genau sieben Tage.

Syntax:

```
LunarWeekName(date [, period_no[, first_week_day]])
```

Rückgabe Datentyp: dual

Beispieldiagramm der Funktion `lunarweekname()`



Die Funktion `lunarweekname()` bestimmt, in welche Mondwoche das Datum fällt, wobei die Wochenzählung am 1. Januar beginnt. Dann wird ein Wert zurückgegeben, der aus `year/weekcount` besteht.

Argumente

Argument	Beschreibung
date	Datum oder Zeitstempel für die Evaluierung.
period_no	**period_no** ist eine ganze Zahl oder eine Formel, die eine ganze Zahl ergibt, wobei 0 für die Mondwoche steht, die **date** enthält. Negative Werte von **period_no** stehen für vorangehende, positive Werte für nachfolgende Mondwochen.
first_week_day	Ein Startwert, der größer oder kleiner als Null sein kann. Dadurch wird der Beginn des Jahres um die angegebene Anzahl an Tagen und/oder den Bruchteil eines Tages verschoben.

Verwendung

Die Funktion `lunarweekname()` ist nützlich, wenn Sie Aggregierungen nach Mondwochen vergleichen möchten. Beispielsweise kann die Funktion verwendet werden, um den Gesamtumsatz von Produkten nach Mondwochen zu bestimmen. Mondwochen sind nützlich, wenn Sie sicherstellen möchten, dass alle Werte in der ersten Woche des Jahres nur Werte ab dem 1. Januar als frühestes Datum umfassen.

Diese Dimensionen können im Ladeskript erstellt werden, indem die Funktion verwendet wird, um ein Feld in einer Master-Kalender-Tabelle zu erstellen. Diese Funktion kann auch direkt als berechnete Dimension in einem Diagramm verwendet werden.

Funktionsbeispiele

Beispiel	**Ergebnis**
`lunarweekname('01/12/2013')`	Gibt 2006/02 zurück.
`lunarweekname('01/12/2013', -1)`	Gibt 2006/01 zurück.
`lunarweekname('01/12/2013', 0, 1)`	Gibt 2006/02 zurück.

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Beispiel 1 – Datum ohne zusätzliche Argumente

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz enthält eine Reihe von Transaktionen für 2022 und wird in eine Tabelle namens `Transactions` geladen.
- Das Datumsfeld wird im Format der Systemvariablen `DateFormat` (MM/TT/JJJJ) bereitgestellt.

- Erstellung eines Felds `lunar_week_name`, das das Jahr und die Wochennummer für die Mondwoche zurückgibt, in der die Transaktionen stattfanden

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        lunarweekname(date) as lunar_week_name
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `lunar_week_name`

Ergebnistabelle

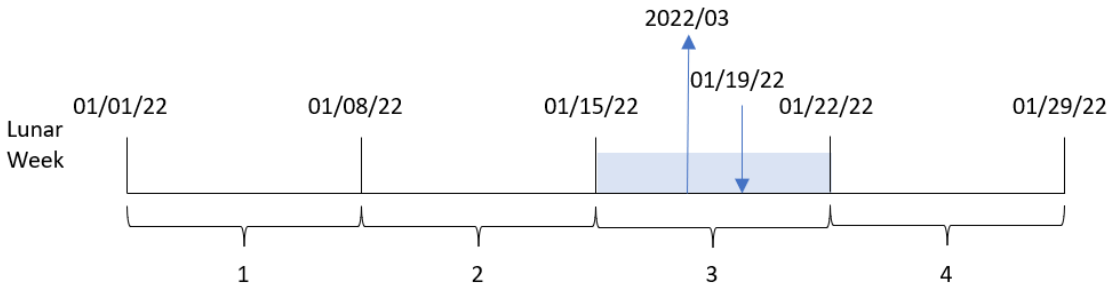
date	lunar_week_name
1/7/2022	2022/01

date	lunar_week_name
1/19/2022	2022/03
2/5/2022	2022/06
2/28/2022	2022/09
3/16/2022	2022/11
4/1/2022	2022/13
5/7/2022	2022/19
5/16/2022	2022/20
6/15/2022	2022/24
6/26/2022	2022/26
7/9/2022	2022/28
7/22/2022	2022/29
7/23/2022	2022/30
7/27/2022	2022/30
8/2/2022	2022/31
8/8/2022	2022/32
8/19/2022	2022/33
9/26/2022	2022/39
10/14/2022	2022/41
10/29/2022	2022/44

Das Feld `lunar_week_name` wird in dem vorangehenden load-Befehl erstellt, indem die Funktion `lunarweekname()` verwendet und das Feld `date` als Argument der Funktion übergeben wird.

Die Funktion `lunarweekname()` identifiziert, in welche Mondwoche der Datumswert fällt, und gibt das Jahr und die Wochennummer für dieses Datum zurück.

Diagramm der Funktion `lunarweekname()`, Beispiel ohne zusätzliche Argumente



Transaktion 8189 fand am 19. Januar statt. Die Funktion `lunarweekname()` identifiziert, dass dieses Datum in die Mondwoche fällt, die am 15. Januar beginnt. Das ist die dritte Mondwoche des Jahres. Daher ist der von dieser Transaktion zurückgegebene Wert für `lunar_week_name` 2022/03.

Beispiel 2 – Datum mit Argument „period_no“

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Derselbe Datensatz und dasselbe Szenario wie im ersten Beispiel.
- Es wird ein Feld `previous_lunar_week_name` erstellt, das das Jahr und die Wochennummer für die Mondwoche vor dem Transaktionsdatum zurückgibt.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        lunarweekname(date,-1) as previous_lunar_week_name
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

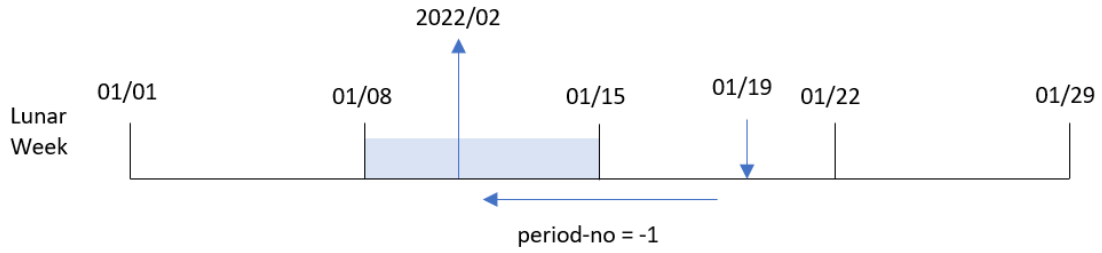
- `date`
- `previous_lunar_week_name`

Ergebnistabelle

date	previous_lunar_week_name
1/7/2022	2021/52
1/19/2022	2022/02
2/5/2022	2022/05
2/28/2022	2022/08
3/16/2022	2022/10
4/1/2022	2022/12
5/7/2022	2022/18
5/16/2022	2022/19
6/15/2022	2022/23
6/26/2022	2022/25
7/9/2022	2022/27
7/22/2022	2022/28
7/23/2022	2022/29
7/27/2022	2022/29
8/2/2022	2022/30
8/8/2022	2022/31
8/19/2022	2022/32
9/26/2022	2022/38
10/14/2022	2022/40
10/29/2022	2022/43

Da in diesem Fall eine `period_no` von -1 als Versatzargument in der Funktion `lunarweekname()` verwendet wurde, identifiziert die Funktion zuerst die Mondwoche, in der die Transaktionen stattfanden. Dann gibt sie das Jahr und die Nummer der davor liegenden Woche zurück.

Diagramm der Funktion `lunarweekname()`, Beispiel „period_no“



Transaktion 8189 fand am 19. Januar statt. Die Funktion `lunarweekname()` identifiziert, dass diese Transaktion in der dritten Mondwoche des Jahres stattfand. Daher gibt sie das Jahr und den Wert für eine Woche davor, 2022/02, für das Feld `previous_lunar_week_name` zurück.

Beispiel 3 – Datum mit dem Argument „first_week_day“

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript verwendet den gleichen Datensatz und das gleiche Szenario wie das erste Beispiel. In diesem Beispiel wird für den Beginn der Mondwochen der 5. Januar festgelegt.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        lunarweekname(date,0,4) as lunar_week_name
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
```

```
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

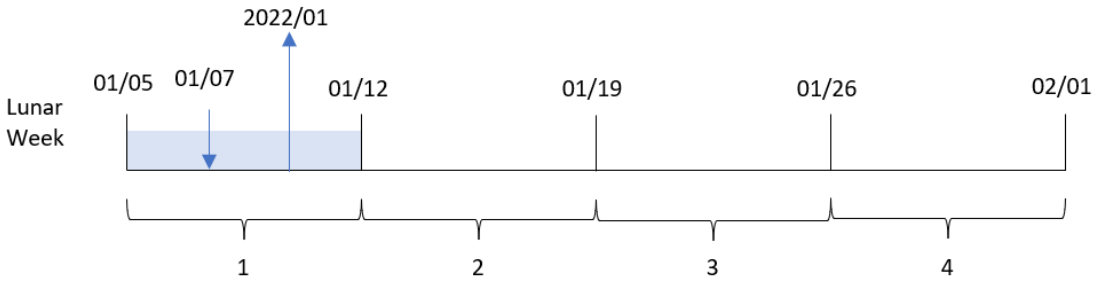
- `date`
- `lunar_week_name`

Ergebnistabelle

date	lunar_week_name
1/7/2022	2022/01
1/19/2022	2022/03
2/5/2022	2022/05
2/28/2022	2022/08
3/16/2022	2022/11
4/1/2022	2022/13
5/7/2022	2022/18
5/16/2022	2022/19
6/15/2022	2022/24
6/26/2022	2022/25
7/9/2022	2022/27
7/22/2022	2022/29
7/23/2022	2022/29
7/27/2022	2022/30
8/2/2022	2022/30
8/8/2022	2022/31
8/19/2022	2022/33
9/26/2022	2022/38

date	lunar_week_name
10/14/2022	2022/41
10/29/2022	2022/43

Diagramm der Funktion `lunarweekname()`, Beispiel „first_week_day“



Da in diesem Fall das Argument `first_week_date` von 4 in der Funktion `lunarweekname()` verwendet wird, wird der Start der Mondwochen vom 1. auf den 5. Januar verschoben.

Die Transaktion 8188 fand am 7. Januar statt. Da die Mondwochen am 5. Januar beginnen, identifiziert die Funktion `lunarweekname()` die Mondwoche des 7. Januar als die erste Mondwoche des Jahres. Daher ist der für diese Transaktion zurückgegebene Wert von `lunar_week_name` 2022/01.

Beispiel 4 – Diagrammobjektbeispiel

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript verwendet den gleichen Datensatz und das gleiche Szenario wie das erste Beispiel.

In diesem Beispiel wird jedoch der unveränderte Datensatz in die Anwendung geladen. Die Berechnung, die die Nummer der Mondwoche und das Jahr zurückgibt, in denen die Transaktionen stattfanden, wird als Kennzahl in einem Diagrammobjekt der Anwendung erstellt.

Ladeskript

```
Transactions:
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
```

```
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu: `date`.

Um das Startdatum der Mondwoche zu berechnen, in dem eine Transaktion stattfindet, erstellen Sie die folgende Kennzahl:

```
=lunarweekname(date)
```

Ergebnistabelle

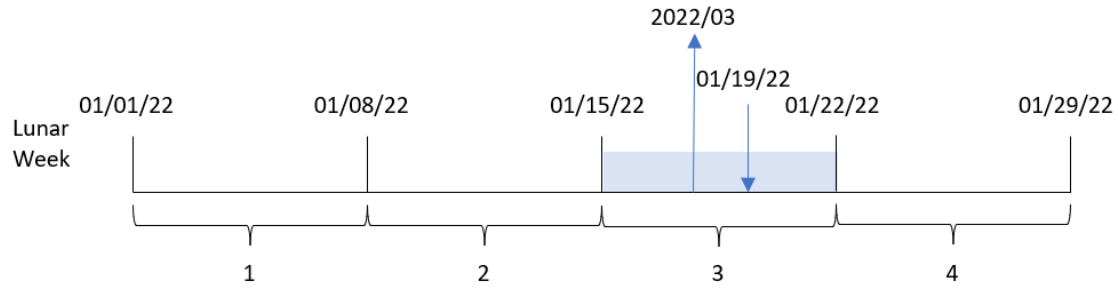
date	=lunarweekname(date)
1/7/2022	2022/01
1/19/2022	2022/03
2/5/2022	2022/06
2/28/2022	2022/09
3/16/2022	2022/11
4/1/2022	2022/13
5/7/2022	2022/19
5/16/2022	2022/20
6/15/2022	2022/24
6/26/2022	2022/26
7/9/2022	2022/28
7/22/2022	2022/29
7/23/2022	2022/30

date	=lunarweekname(date)
7/27/2022	2022/30
8/2/2022	2022/31
8/8/2022	2022/32
8/19/2022	2022/33
9/26/2022	2022/39
10/14/2022	2022/41
10/29/2022	2022/44

Die Kennzahl `lunar_week_name` wird im Diagrammobjekt erstellt, indem die Funktion `lunarweekname()` verwendet und das Feld `date` als Argument der Funktion übergeben wird.

Die Funktion `lunarweekname()` identifiziert, in welche Mondwoche der Datumswert fällt, und gibt das Jahr und die Wochennummer für dieses Datum zurück.

Diagramm der Funktion `lunarweekname()`, Diagrammobjektbeispiel



Transaktion 8189 fand am 19. Januar statt. Die Funktion `lunarweekname()` identifiziert, dass dieses Datum in die Mondwoche fällt, die am 15. Januar beginnt. Das ist die dritte Mondwoche des Jahres. Daher ist der Wert von `lunar_week_name` für diese Transaktion 2022/03.

Beispiel 5 – Szenario

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz enthält eine Reihe von Transaktionen für 2022 und wird in eine Tabelle namens `Transactions` geladen.
- Datumsfeld, das im Format der Systemvariablen `DateFormat` (MM/TT/JJJJ) bereitgestellt wird

Der Endbenutzer möchte ein Diagrammobjekt, das den Gesamtumsatz nach Woche für das laufende Jahr darstellt. Woche 1 mit einer Länge von sieben Tagen soll am 1. Januar beginnen. Das kann auch dann erreicht werden, wenn diese Dimension nicht im Datenmodell verfügbar ist, indem die Funktion `lunarweekname()` als berechnete Dimension im Diagramm verwendet wird.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Ergebnisse

Gehen Sie folgendermaßen vor:

1. Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle.
2. Erstellen Sie eine berechnete Dimension anhand der folgenden Formel:
   `=lunarweekname(date)`
3. Berechnen Sie den Gesamtumsatz anhand der folgenden Aggregierungskennzahl:
   `=sum(amount)`
4. Legen Sie das **Zahlenformat** der Kennzahl auf **Währung** fest.

Ergebnistabelle

=lunarweekname(date)	=sum(amount)
2022/01	$17.17
2022/03	$37.23
2022/06	$57.42
2022/09	$88.27
2022/11	$53.80
2022/13	$82.06
2022/19	$40.39
2022/20	$87.21
2022/24	$95.93
2022/26	$45.89
2022/28	$36.23
2022/29	$25.66
2022/30	$152.75
2022/31	$76.11
2022/32	$25.12
2022/33	$46.23
2022/39	$84.21
2022/41	$96.24
2022/44	$67.67

lunarweekstart

Diese Funktion liefert einen Wert, der dem Zeitstempel der ersten Millisekunde des ersten Tags der Mondwoche entspricht, in der **date** liegt. Bei Mondwochen in Qlik Sense wird der 1. Januar als der erste Tag der Woche gezählt. Mit Ausnahme der letzten Woche des Jahres umfasst jede Woche genau sieben Tage.

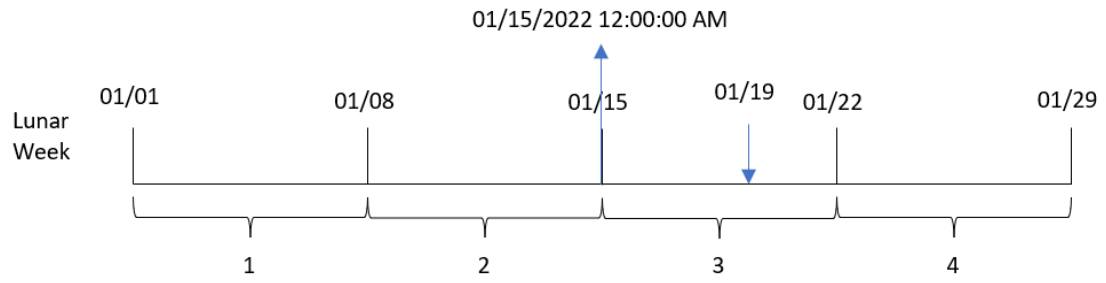
Syntax:

```
LunarweekStart(date[, period_no[, first_week_day]])
```

Rückgabe Datentyp: dual

Die Funktion `lunarweekstart()` bestimmt, in welche Mondwoche `date` fällt. Sie gibt dann einen Zeitstempel im Datumsformat für die erste Millisekunde dieser Woche zurück.

Beispieldiagramm der Funktion `lunarweekstart()`



Argumente

Argument	Beschreibung
date	Datum oder Zeitstempel für die Evaluierung.
period_no	**period_no** ist eine ganze Zahl oder eine Formel, die eine ganze Zahl ergibt, wobei 0 für die Mondwoche steht, die **date** enthält. Negative Werte von **period_no** stehen für vorangehende, positive Werte für nachfolgende Mondwochen.
first_week_day	Ein Startwert, der größer oder kleiner als Null sein kann. Dadurch wird der Beginn des Jahres um die angegebene Anzahl an Tagen und/oder den Bruchteil eines Tages verschoben.

Verwendung

Die Funktion `lunarweekstart()` wird in der Regel als Teil einer Formel verwendet, wenn in der Berechnung der Teil der Woche verwendet werden soll, der bereits verstrichen ist. Anders als bei der Funktion `weekstart()` beginnen beim Start eines neuen Kalenderjahres die Wochen am 1. Januar, und jede nachfolgende Woche beginnt sieben Tage später. Die Funktion `lunarweekstart()` ist nicht von der Systemvariablen `FirstWeekDay` betroffen.

Zum Beispiel kann `lunarweekstart()` verwendet werden, um die Zinsen zu berechnen, die in einem Jahr bis dato aufgelaufen sind.

Funktionsbeispiele

Beispiel	Ergebnis
`lunarweekstart ('01/12/2013')`	Gibt 01/08/2013 zurück.
`lunarweekstart ('01/12/2013', -1)`	Gibt 01/01/2013 zurück.
`lunarweekstart ('01/12/2013', 0, 1 )`	Gibt 01/09/2013 zurück, weil das Festlegen von first_week_day auf 1 bedeutet, dass der Anfang des Jahres auf den 01/02/2013 geändert wird.

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Beispiel 1 – keine zusätzlichen Argumente

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz enthält eine Reihe von Transaktionen für 2022 und wird in eine Tabelle namens `Transactions` geladen.
- Das Datumsfeld wird im Format der Systemvariablen `DateFormat` (MM/TT/JJJJ) bereitgestellt.
- Erstellung eines Felds `start_of_week`, das den Zeitstempel für den Start der Mondwoche zurückgibt, in der die Transaktionen stattfanden.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        lunarweekstart(date) as start_of_week,
        timestamp(lunarweekstart(date)) as start_of_week_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
```

```
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `start_of_week`
- `start_of_week_timestamp`

Ergebnistabelle

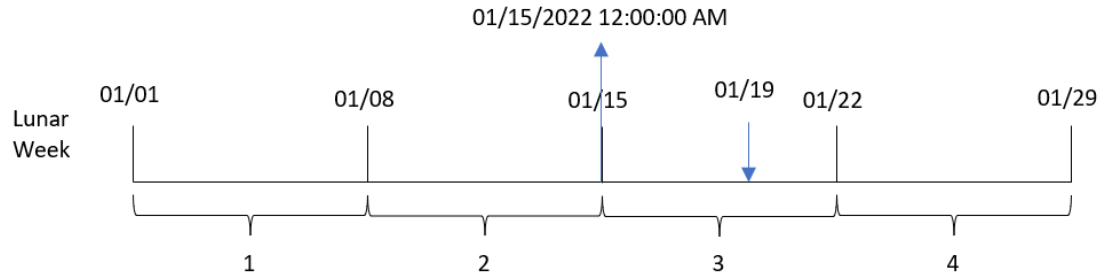
date	start_of_week	start_of_week_timestamp
1/7/2022	01/01/2022	1/1/2022 12:00:00 AM
1/19/2022	01/15/2022	1/15/2022 12:00:00 AM
2/5/2022	02/05/2022	2/5/2022 12:00:00 AM
2/28/2022	02/26/2022	2/26/2022 12:00:00 AM
3/16/2022	03/12/2022	3/12/2022 12:00:00 AM
4/1/2022	03/26/2022	3/26/2022 12:00:00 AM
5/7/2022	05/07/2022	5/7/2022 12:00:00 AM
5/16/2022	05/14/2022	5/14/2022 12:00:00 AM
6/15/2022	06/11/2022	6/11/2022 12:00:00 AM
6/26/2022	06/25/2022	6/25/2022 12:00:00 AM
7/9/2022	07/09/2022	7/9/2022 12:00:00 AM
7/22/2022	07/16/2022	7/16/2022 12:00:00 AM

date	start_of_week	start_of_week_timestamp
7/23/2022	07/23/2022	7/23/2022 12:00:00 AM
7/27/2022	07/23/2022	7/23/2022 12:00:00 AM
8/2/2022	07/30/2022	7/30/2022 12:00:00 AM
8/8/2022	08/06/2022	8/6/2022 12:00:00 AM
8/19/2022	08/13/2022	8/13/2022 12:00:00 AM
9/26/2022	09/24/2022	9/24/2022 12:00:00 AM
10/14/2022	10/08/2022	10/8/2022 12:00:00 AM
10/29/2022	10/29/2022	10/29/2022 12:00:00 AM

Das Feld `start_of_week` wird im vorangehenden load-Befehl erstellt, indem die Funktion `lunarweekstart()` verwendet und das Feld `date` als Argument der Funktion übergeben wird.

Die Funktion `lunarweekstart()` identifiziert, in welche Mondwoche das Datum fällt, und gibt einen Zeitstempel für die erste Millisekunde dieser Woche zurück.

Diagramm der Funktion `lunarweekstart()`, Beispiel ohne zusätzliche Argumente



Transaktion 8189 fand am 19. Januar statt. Die Funktion `lunarweekstart()` identifiziert, dass die Mondwoche am 15. Januar beginnt. Daher gibt der Wert für `start_of_week` dieser Transaktion die erste Millisekunde dieses Tages zurück, also den 15. Januar um 12:00:00 AM.

Beispiel 2 – period_no

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Derselbe Datensatz und dasselbe Szenario wie im ersten Beispiel.
- Es wird ein Feld `previous_lunar_week_start` erstellt, das den Zeitstempel für den Start der Mondwoche vor der Woche zurückgibt, in der die Transaktion stattfand.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        lunarweekstart(date,-1) as previous_lunar_week_start,
        timestamp(lunarweekstart(date,-1)) as previous_lunar_week_start_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Ergebnisse

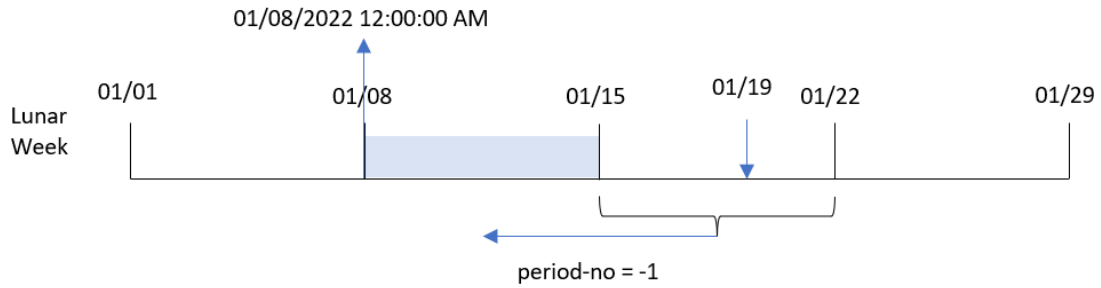
Ergebnistabelle

date	previous_lunar_week_start	previous_lunar_week_start_timestamp
1/7/2022	12/24/2021	12/24/2021 12:00:00 AM
1/19/2022	01/08/2022	1/8/2022 12:00:00 AM
2/5/2022	01/29/2022	1/29/2022 12:00:00 AM

date	previous_lunar_week_start	previous_lunar_week_start_timestamp
2/28/2022	02/19/2022	2/19/2022 12:00:00 AM
3/16/2022	03/05/2022	3/5/2022 12:00:00 AM
4/1/2022	03/19/2022	3/19/2022 12:00:00 AM
5/7/2022	04/30/2022	4/30/2022 12:00:00 AM
5/16/2022	05/07/2022	5/7/2022 12:00:00 AM
6/15/2022	06/04/2022	6/4/2022 12:00:00 AM
6/26/2022	06/18/2022	6/18/2022 12:00:00 AM
7/9/2022	07/02/2022	7/2/2022 12:00:00 AM
7/22/2022	07/09/2022	7/9/2022 12:00:00 AM
7/23/2022	07/16/2022	7/16/2022 12:00:00 AM
7/27/2022	07/16/2022	7/16/2022 12:00:00 AM
8/2/2022	07/23/2022	7/23/2022 12:00:00 AM
8/8/2022	07/30/2022	7/30/2022 12:00:00 AM
8/19/2022	08/06/2022	8/6/2022 12:00:00 AM
9/26/2022	09/17/2022	9/17/2022 12:00:00 AM
10/14/2022	10/01/2022	10/1/2022 12:00:00 AM
10/29/2022	10/22/2022	10/22/2022 12:00:00 AM

Da in diesem Fall eine `period_no` von -1 als Versatzargument in der Funktion `lunarweekstart()` verwendet wurde, identifiziert die Funktion zuerst die Mondwoche, in der die Transaktionen stattfinden. Dann geht sie eine Woche zurück und identifiziert die erste Millisekunde dieser Mondwoche.

Diagramm der Funktion `lunarweekstart()`, Beispiel „period_no“



Transaktion 8189 fand am 19. Januar statt. Die Funktion `lunarweekstart()` identifiziert, dass die Mondwoche am 15. Januar beginnt. Daher begann die vorherige Mondwoche am 8. Januar um 12:00:00 AM; dies ist der Wert, der für das Feld `previous_lunar_week_start` zurückgegeben wird.

Beispiel 3 – first_week_day

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript verwendet den gleichen Datensatz und das gleiche Szenario wie das erste Beispiel. In diesem Beispiel wird für den Beginn der Mondwochen der 5. Januar festgelegt.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        lunarweekstart(date,0,4) as start_of_week,
        timestamp(lunarweekstart(date,0,4)) as start_of_week_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

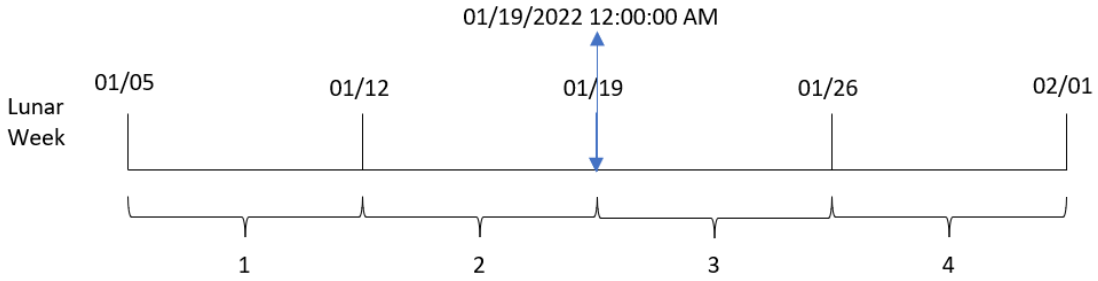
- date
- start_of_week
- start_of_week_timestamp

Ergebnistabelle

date	start_of_week	start_of_week_timestamp
1/7/2022	01/05/2022	1/5/2022 12:00:00 AM
1/19/2022	01/19/2022	1/19/2022 12:00:00 AM
2/5/2022	02/02/2022	2/2/2022 12:00:00 AM
2/28/2022	02/23/2022	2/23/2022 12:00:00 AM
3/16/2022	03/16/2022	3/16/2022 12:00:00 AM
4/1/2022	03/30/2022	3/30/2022 12:00:00 AM
5/7/2022	05/04/2022	5/4/2022 12:00:00 AM
5/16/2022	05/11/2022	5/11/2022 12:00:00 AM
6/15/2022	06/15/2022	6/15/2022 12:00:00 AM
6/26/2022	06/22/2022	6/22/2022 12:00:00 AM
7/9/2022	07/06/2022	7/6/2022 12:00:00 AM
7/22/2022	07/20/2022	7/20/2022 12:00:00 AM
7/23/2022	07/20/2022	7/20/2022 12:00:00 AM
7/27/2022	07/27/2022	7/27/2022 12:00:00 AM
8/2/2022	07/27/2022	7/27/2022 12:00:00 AM
8/8/2022	08/03/2022	8/3/2022 12:00:00 AM
8/19/2022	08/17/2022	8/17/2022 12:00:00 AM
9/26/2022	09/21/2022	9/21/2022 12:00:00 AM
10/14/2022	10/12/2022	10/12/2022 12:00:00 AM
10/29/2022	10/26/2022	10/26/2022 12:00:00 AM

Da in diesem Fall das Argument `first_week_date` von 4 in der Funktion `lunarweekstart()` verwendet wird, wird der Start des Jahres vom 1. auf den 5. Januar verschoben.

Diagramm der Funktion `lunarweekstart()`, Beispiel „first_week_day“



Transaktion 8189 fand am 19. Januar statt. Da Mondwochen am 5. Januar beginnen, identifiziert die Funktion `lunarweekstart()`, dass die Mondwoche, die den 19. Januar enthält, auch am 19. Januar um 12:00:00 AM beginnt. Daher ist dies der Wert, der für das Feld `start_of_week` zurückgegeben wird.

Beispiel 4 – Diagrammobjektbeispiel

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript verwendet den gleichen Datensatz und das gleiche Szenario wie das erste Beispiel.

In diesem Beispiel wird jedoch der unveränderte Datensatz in die Anwendung geladen. Die Berechnung, die einen Zeitstempel für den Start der Mondwoche zurückgibt, in der die Transaktionen stattfanden, wird als Kennzahl in einem Diagrammobjekt der Anwendung erstellt.

Ladeskript

```
Transactions:
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
```

```
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu: `date`.

Fügen Sie die folgenden Kennzahlen hinzu:

```
=lunarweekstart(date)
```

```
=timestamp(lunarweekstart(date))
```

Ergebnistabelle

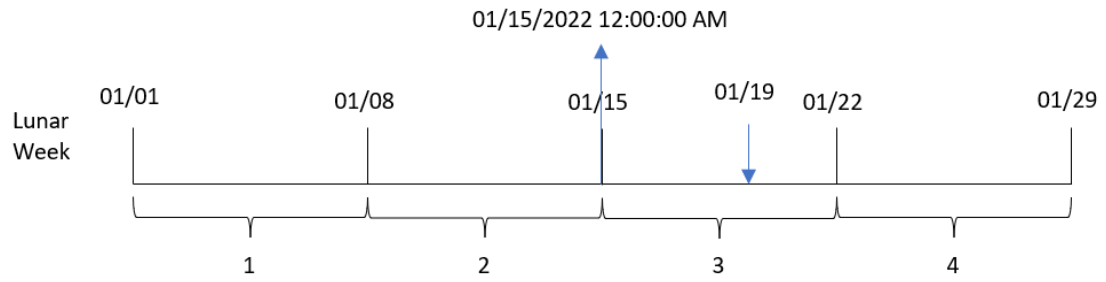
date	=lunarweekstart(date)	=timestamp(lunarweekstart(date))
1/7/2022	01/01/2022	1/1/2022 12:00:00 AM
1/19/2022	01/15/2022	1/15/2022 12:00:00 AM
2/5/2022	02/05/2022	2/5/2022 12:00:00 AM
2/28/2022	02/26/2022	2/26/2022 12:00:00 AM
3/16/2022	03/12/2022	3/12/2022 12:00:00 AM
4/1/2022	03/26/2022	3/26/2022 12:00:00 AM
5/7/2022	05/07/2022	5/7/2022 12:00:00 AM
5/16/2022	05/14/2022	5/14/2022 12:00:00 AM
6/15/2022	06/11/2022	6/11/2022 12:00:00 AM
6/26/2022	06/25/2022	6/25/2022 12:00:00 AM
7/9/2022	07/09/2022	7/9/2022 12:00:00 AM
7/22/2022	07/16/2022	7/16/2022 12:00:00 AM
7/23/2022	07/23/2022	7/23/2022 12:00:00 AM
7/27/2022	07/23/2022	7/23/2022 12:00:00 AM
8/2/2022	07/30/2022	7/30/2022 12:00:00 AM
8/8/2022	08/06/2022	8/6/2022 12:00:00 AM
8/19/2022	08/13/2022	8/13/2022 12:00:00 AM
9/26/2022	09/24/2022	9/24/2022 12:00:00 AM

date	=lunarweekstart(date)	=timestamp(lunarweekstart(date))
10/14/2022	10/08/2022	10/8/2022 12:00:00 AM
10/29/2022	10/29/2022	10/29/2022 12:00:00 AM

Die Kennzahl `start_of_week` wird im Diagrammobjekt erstellt, indem die Funktion `lunarweekstart()` verwendet und das Datumsfeld als Argument der Funktion übergeben wird.

Die Funktion `lunarweekstart()` identifiziert, in welche Mondwoche der Datumswert fällt, und gibt einen Zeitstempel für die letzte Millisekunde dieser Woche zurück.

Diagramm der Funktion `lunarweekstart()`, Diagrammobjektbeispiel



Transaktion 8189 fand am 19. Januar statt. Die Funktion `lunarweekstart()` identifiziert, dass die Mondwoche am 15. Januar beginnt. Daher ist der Wert für `start_of_week` dieser Transaktion die erste Millisekunde dieses Tages, also der 15. Januar um 12:00:00 AM.

Beispiel 5 – Szenario

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Datensatz mit einer Reihe von Darlehenssalden, der in eine Tabelle namens `Loans` geladen wird
- Daten aus der Darlehens-ID, dem Saldo zum Wochenbeginn und dem einfachen Zinssatz, der für jedes Darlehen pro Jahr berechnet wird

Der Endbenutzer möchte ein Diagrammobjekt, das nach Darlehens-ID die aktuellen Zinsen anzeigt, die für jedes Darlehen in der Woche bis dato aufgelaufen sind.

Ladeskript

```
Loans:
Load
*
Inline
```

```
[
loan_id,start_balance,rate
8188,$10000.00,0.024
8189,$15000.00,0.057
8190,$17500.00,0.024
8191,$21000.00,0.034
8192,$90000.00,0.084
];
```

Ergebnisse

Gehen Sie folgendermaßen vor:

1. Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle.
2. Fügen Sie die folgenden Felder als Dimensionen hinzu:
   - `loan_id`
   - `start_balance`
3. Erstellen Sie dann die folgende Kennzahl, um die kumulierten Zinsen zu berechnen: `=start_balance*(rate*(today(1)-lunarweekstart(today(1)))/365)`
4. Legen Sie das **Zahlenformat** der Kennzahl auf **Währung** fest.

Ergebnistabelle

loan_id	start_balance	=start_balance*(rate*(today(1)- lunarweekstart (today(1)))/365)
8188	$10000.00	$15.07
8189	$15000.00	$128.84
8190	$17500.00	$63.29
8191	$21000.00	$107.59
8192	$90000.00	$1139.18

Die Funktion `lunarweekstart()` verwendet das aktuelle Datum als einziges Argument und gibt das Startdatum des aktuellen Jahres zurück. Die Formel zieht dieses Ergebnis vom aktuellen Datum ab und gibt die Anzahl der Tage zurück, die bisher in der Woche verstrichen sind.

Dieser Wert wird dann mit dem Zinssatz multipliziert und durch 365 geteilt, um den effektiven Zinssatz für diesen Zeitraum zurückzugeben. Das Ergebnis wird dann mit dem Anfangssaldo des Darlehens multipliziert, was die Zinsen ergibt, die bislang in dieser Woche aufgelaufen sind.

makedate

Die Funktion liefert ein Datum bestehend aus der angegebenen Jahreszahl **YYYY**, dem Monat **MM** und dem Wochentag **DD**.

Syntax:

```
MakeDate(YYYY [ , MM [ , DD ] ])
```

Rückgabe Datentyp: dual

Argumente

Argument	Beschreibung
YYYY	Das Jahr als ganze Zahl.
MM	Der Monat als ganze Zahl. Ist kein Monat angegeben, wird 1 (Januar) angenommen.
DD	Der Tag als ganze Zahl. Ist kein Tag angegeben, wird 1 (1.) angenommen.

Verwendung

Die Funktion `makedate()` wird in der Regel im Skript zur Datengenerierung für die Generierung eines Kalenders verwendet. Sie kann auch verwendet werden, wenn das Datumsfeld nicht direkt als Datum verfügbar ist, sondern Umwandlungen benötigt, um Jahres-, Monats- und Tageskomponenten zu extrahieren.

In diesen Beispielen wird das Datumsformat MM/TT/JJJJ verwendet. Das Datumsformat wird im Befehl `SET DateFormat` oben in Ihrem Datenladeskript angegeben. Ändern Sie das Format in den Beispielen entsprechend Ihren Anforderungen.

Funktionsbeispiele

Beispiel	Ergebnis
`makedate(2012)`	Gibt 01/01/2012 zurück.
`makedate(12)`	Gibt 01/01/2012 zurück.
`makedate(2012,12)`	Gibt 12/01/2012 zurück.
`makedate(2012,2,14)`	Gibt 02/14/2012 zurück.

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Beispiel 1 – einfaches Beispiel

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz enthält eine Reihe von Transaktionen für 2018 und wird in eine Tabelle namens `Transactions` geladen.
- Das Datumsfeld wird im Format der Systemvariablen `DateFormat` (MM/TT/JJJJ) bereitgestellt.
- Erstellung eines Felds `transaction_date`, das ein Datum im Format MM/TT/JJJJ zurückgibt

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        makedate(transaction_year, transaction_month, transaction_day) as transaction_date
        ;
Load * Inline [
transaction_id, transaction_year, transaction_month, transaction_day, transaction_amount,
transaction_quantity, customer_id
3750, 2018, 08, 30, 12423.56, 23, 2038593
3751, 2018, 09, 07, 5356.31, 6, 203521
3752, 2018, 09, 16, 15.75, 1, 5646471
3753, 2018, 09, 22, 1251, 7, 3036491
3754, 2018, 09, 22, 21484.21, 1356, 049681
3756, 2018, 09, 22, -59.18, 2, 2038593
3757, 2018, 09, 23, 3177.4, 21, 203521
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `transaction_year`
- `transaction_month`
- `transaction_day`
- `transaction_date`

Ergebnistabelle

transaction_year	transaction_month	transaction_day	transaction_date
2018	08	30	08/30/2018
2018	09	07	09/07/2018
2018	09	16	09/16/2018
2018	09	22	09/22/2018
2018	09	23	09/23/2018

Das Feld `transaction_date` wird im vorangehenden load-Befehl erstellt, indem die Funktion `makedate()` verwendet und die Felder für Jahr, Monat und Tag als Argumente der Funktion übergeben werden.

Die Funktion kombiniert diese Werte dann und konvertiert sie in ein Datumsfeld. Die Ergebnisse werden im Format der Systemvariablen `DateFormat` zurückgegeben.

Beispiel 2 – Geändertes „DateFormat“

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Derselbe Datensatz und dasselbe Szenario wie im ersten Beispiel.
- Es wird ein Feld `transaction_date` im Format TT/MM/JJJJ erstellt, ohne die Systemvariable `DateFormat` zu ändern.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        date(makedate(transaction_year, transaction_month, transaction_day), 'DD/MM/YYYY') as
transaction_date
        ;
Load * Inline [
transaction_id, transaction_year, transaction_month, transaction_day, transaction_amount,
transaction_quantity, customer_id
3750, 2018, 08, 30, 12423.56, 23, 2038593
3751, 2018, 09, 07, 5356.31, 6, 203521
3752, 2018, 09, 16, 15.75, 1, 5646471
3753, 2018, 09, 22, 1251, 7, 3036491
3754, 2018, 09, 22, 21484.21, 1356, 049681
3756, 2018, 09, 22, -59.18, 2, 2038593
```

```
3757, 2018, 09, 23, 3177.4, 21, 203521
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `transaction_year`
- `transaction_month`
- `transaction_day`
- `transaction_date`

Ergebnistabelle

transaction_year	transaction_month	transaction_day	transaction_date
2018	08	30	30/08/2018
2018	09	07	07/09/2018
2018	09	16	16/09/2018
2018	09	22	22/09/2018
2018	09	23	23/09/2018

In diesem Fall ist die Funktion `makedate()` innerhalb der Funktion `date()` verschachtelt. Das zweite Argument der Funktion `date()` legt das Format der Funktionsergebnisse von `makedate()` auf das erforderliche TT/MM/JJJJ fest.

Beispiel 3 – Diagrammobjektbeispiel

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Datensatz mit einer Reihe von Transaktionen für 2018, der in eine Tabelle namens `Transactions` geladen wird
- Transaktionsdatumswerte, die in zwei Feldern bereitgestellt werden: `year` und `month`

Erstellung einer Diagrammobjektkennzahl, `transaction_date`, die ein Datum im Format MM/TT/JJJJ zurückgibt

Ladeskript

```
SET DateFormat='MM/DD/YYYY';
```

```
Transactions:
Load * Inline [
transaction_id, transaction_year, transaction_month, transaction_amount, transaction_quantity,
customer_id
3750, 2018, 08, 12423.56, 23, 2038593
3751, 2018, 09, 5356.31, 6, 203521
3752, 2018, 09, 15.75, 1, 5646471
3753, 2018, 09, 1251, 7, 3036491
3754, 2018, 09, 21484.21, 1356, 049681
3756, 2018, 09, -59.18, 2, 2038593
3757, 2018, 09, 3177.4, 21, 203521
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `year`
- `month`

Um das `transaction_date` zu bestimmen, erstellen Sie die folgende Kennzahl:

```
=makedate(transaction_year,transaction_month)
```

Ergebnistabelle

transaction_year	transaction_month	transaction_date
2018	08	08/01/2018
2018	09	09/01/2018

Die Kennzahl `transaction_date` wird im Diagrammobjekt erstellt, indem die Funktion `makedate()` verwendet und die Felder für Jahr und Monat als Argumente der Funktion übergeben werden.

Die Funktion kombiniert dann diese Werte, ebenso wie den angenommenen Tageswert von 01. Diese Werte werden dann in ein Datumsfeld konvertiert. Die Ergebnisse werden im Format der Systemvariablen `DateFormat` zurückgegeben.

Beispiel 4 – Szenario

Ladeskript und Diagrammformel

Übersicht

Erstellen Sie einen Kalenderdatensatz für das Kalenderjahr 2022.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Calendar:
```

```
       load
              *
    where year(date)=2022;
load
       date(recno()+makedate(2021,12,31)) as date
AutoGenerate 400;
```

Ergebnisse

Ergebnistabelle

date
01/01/2022
01/02/2022
01/03/2022
01/04/2022
01/05/2022
01/06/2022
01/07/2022
01/08/2022
01/09/2022
01/10/2022
01/11/2022
01/12/2022
01/13/2022
01/14/2022
01/15/2022
01/16/2022
01/17/2022
01/18/2022
01/19/2022
01/20/2022
01/21/2022
01/22/2022
01/23/2022
01/24/2022

date
01/25/2022
+ 340 weitere Zeilen

Die Funktion `makedate()` erstellt einen Datumswert für den 31. Dezember 2021. Die Funktion `recno()` stellt die Datensatznummer des aktuellen Datensatzes bereit, der in die Tabelle geladen wird, beginnend mit 1. Daher hat der erste Datensatz das Datum 1. Januar 2022. Für jede darauffolgende `recno()` wird dieses Datum um 1 erhöht. Diese Formel wird in eine Funktion `date()` eingeschlossen, um den Wert in ein Datum zu konvertieren. Dieser Vorgang wird von der Funktion `autogenerate` 400 Mal wiederholt. Abschließend kann mit einer vorangehenden load-Anweisung eine `where`-Bedingung verwendet werden, um nur Datumswerte des Jahres 2022 zu laden. Mir diesem Skript wird ein Kalender generiert, der jedes Datum im Jahr 2022 enthält.

maketime

Die Funktion liefert eine Zeit bestehend aus der angegebenen Stunde **hh**, der Minute **mm** und der Sekunde **ss**.

Syntax:

```
MakeTime(hh [ , mm [ , ss ] ])
```

Rückgabe Datentyp: dual

Argumente

Argument	Beschreibung
hh	Die Stunde als ganze Zahl.
mm	Die Minute als ganze Zahl. Fehlt die Zahl der Minuten, wird 00 angenommen.
ss	Die Sekunde als ganze Zahl. Fehlt die Zahl der Sekunden, wird 00 angenommen.

Verwendung

Die Funktion `maketime()` wird in der Regel im Skript zur Datengenerierung für die Generierung eines Uhrzeitfelds verwendet. Wenn das Uhrzeitfeld aus einem Eingabefeld abgeleitet wird, kann diese Funktion verwendet werden, um die Uhrzeit anhand ihrer Komponenten zu konstruieren.

In diesen Beispielen wird das Uhrzeitformat `h:mm:ss` verwendet. Das Uhrzeitformat wird im Befehl `SET TimeFormat` oben in Ihrem Datenladeskript angegeben. Ändern Sie das Format in den Beispielen entsprechend Ihren Anforderungen.

Funktionsbeispiele

Beispiel	Ergebnis
`maketime(22)`	Gibt 22:00:00 zurück.

Beispiel	Ergebnis
maketime(22, 17)	Gibt 22:17:00 zurück.
maketime(22,17,52)	Gibt 22:17:52 zurück.

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Beispiel 1 – maketime()

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Datensatz mit einer Reihe von Transaktionen, der in eine Tabelle namens `Transactions` geladen wird
- Transaktionszeiten, die in drei Feldern bereitgestellt werden: `hours`, `minutes` und `seconds`.
- Erstellung eines Felds `transaction_time`, das die Uhrzeit im Format der Systemvariablen `TimeFormat` zurückgibt.

Ladeskript

```
SET TimeFormat='h:mm:ss TT';

Transactions:
    Load
        *,
        maketime(transaction_hour, transaction_minute, transaction_second) as transaction_time
        ;
Load * Inline [
transaction_id, transaction_hour, transaction_minute, transaction_second, transaction_amount,
transaction_quantity, customer_id
3750, 18, 43, 30, 12423.56, 23, 2038593
3751, 6, 32, 07, 5356.31, 6, 203521
```

```
3752, 12, 09, 16, 15.75, 1, 5646471
3753, 21, 43, 41, 7, 3036491
3754, 17, 55, 22, 21484.21, 1356, 049681
3756, 2, 52, 22, -59.18, 2, 2038593
3757, 9, 25, 23, 3177.4, 21, 203521
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `transaction_hour`
- `transaction_minute`
- `transaction_second`
- `transaction_time`

Ergebnistabelle

transaction_hour	transaction_minute	transaction_second	transaction_time
2	52	22	2:52:22 AM
6	32	07	6:32:07 AM
9	25	23	9:25:23 AM
12	09	16	12:09:16 PM
17	55	22	5:55:22 PM
18	43	30	6:43:30 PM
21	43	41	9:43:41 PM

Das Feld `transaction_time` wird im vorangehenden load-Befehl erstellt, indem die Funktion `maketime()` verwendet und die Felder für Stunde, Minute und Sekunde als Argumente der Funktion übergeben werden.

Die Funktion kombiniert diese Werte dann und konvertiert sie in ein Uhrzeitfeld. Die Ergebnisse werden im Uhrzeitformat der Systemvariablen `TimeFormat` zurückgegeben.

Beispiel 2 – Funktion „time()“

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Derselbe Datensatz und dasselbe Szenario wie im ersten Beispiel.
- Es wird ein Feld `transaction_time` erstellt, mit dem die Ergebnisse im 24-Stunden-Uhrzeitformat angezeigt werden können, ohne die Systemvariable `TimeFormat` zu verändern.

Ladeskript

```
SET TimeFormat='h:mm:ss TT';

Transactions:
    Load
        *,
        time(maketime(transaction_hour, transaction_minute, transaction_second),'h:mm:ss') as
transaction_time
        ;
Load * Inline [
transaction_id, transaction_hour, transaction_minute, transaction_second, transaction_amount,
transaction_quantity, customer_id
3750, 18, 43, 30, 12423.56, 23, 2038593
3751, 6, 32, 07, 5356.31, 6, 203521
3752, 12, 09, 16, 15.75, 1, 5646471
3753, 21, 43, 41, 7, 3036491
3754, 17, 55, 22, 21484.21, 1356, 049681
3756, 2, 52, 22, -59.18, 2, 2038593
3757, 9, 25, 23, 3177.4, 21, 203521
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `transaction_hour`
- `transaction_minute`
- `transaction_second`
- `transaction_time`

Ergebnistabelle

transaction_hour	transaction_minute	transaction_second	transaction_time
2	52	22	2:52:22
6	32	07	6:32:07
9	25	23	9:25:23
12	09	16	12:09:16
17	55	22	17:55:22
18	43	30	18:43:30
21	43	41	21:43:41

In diesem Fall ist die Funktion `maketime()` innerhalb der Funktion `time()` verschachtelt. Das zweite Argument der Funktion `time()` legt das Format der Funktionsergebnisse von `maketime()` auf das erforderliche `h:mm:ss` fest.

Beispiel 3 – Diagrammobjektbeispiel

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Datensatz mit einer Reihe von Transaktionen, der in eine Tabelle namens `Transactions` geladen wird
- Transaktionszeiten, die in zwei Feldern bereitgestellt werden: `hours` und `minutes`
- Erstellung eines Felds `transaction_time`, das die Uhrzeit im Format der Systemvariablen `TimeFormat` zurückgibt

Erstellen Sie eine Diagrammobjektkennzahl,`transaction_time` , die eine Uhrzeit im Format `h:mm:ss TT` zurückgibt.

Ladeskript

```
SET TimeFormat='h:mm:ss TT';

Transactions:
Load * Inline [
transaction_id, transaction_hour, transaction_minute, transaction_amount, transaction_
quantity, customer_id
3750, 18, 43, 12423.56, 23, 2038593
3751, 6, 32, 5356.31, 6, 203521
3752, 12, 09, 15.75, 1, 5646471
3753, 21, 43, 7, 3036491
3754, 17, 55, 21484.21, 1356, 049681
3756, 2, 52, -59.18, 2, 2038593
3757, 9, 25, 3177.4, 21, 203521
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `transaction_hour`
- `transaction_minute`

Um die `transaction_time` zu berechnen, erstellen Sie die folgende Kennzahl:

```
=maketime(transaction_hour,transaction_minute)
```

Ergebnistabelle

transaction_hour	transaction_minute	=maketime(transaction_hour, transaction_minute)
2	52	2:52:00 AM
6	32	6:32:00 AM
9	25	9:25:00 AM
12	09	12:09:00 PM
17	55	5:55:00 PM
18	43	6:43:00 PM
21	43	9:43:00 PM

Die Kennzahl `transaction_time` wird im Diagrammobjekt erstellt, indem die Funktion `maketime()` verwendet und die Felder für Stunde und Minute als Argumente der Funktion übergeben werden.

Die Funktion kombiniert dann diese Werte, und für Sekunden wird 00 angenommen. Diese Werte werden dann in ein Uhrzeitfeld konvertiert. Die Ergebnisse werden im Format der Systemvariablen `TimeFormat` zurückgegeben.

Beispiel 4 – Szenario

Ladeskript und Diagrammformel

Übersicht

Erstellen Sie einen Kalenderdatensatz für den Monat Januar 2022, unterteilt in 8-Stunden-Inkremente.

Ladeskript

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

tmpCalendar:
       load
               *
    where year(date)=2022;
load
       date(recno()+makedate(2021,12,31)) as date
AutoGenerate 31;

Left join(tmpCalendar)
load
       maketime((recno()-1)*8,00,00) as time
autogenerate 3;

Calendar:
load
       timestamp(date + time) as timestamp
resident tmpCalendar;
```

```
drop table tmpCalendar;
```

Ergebnisse

Ergebnistabelle

Zeitstempel
1/1/2022 12:00:00 AM
1/1/2022 8:00:00 AM
1/1/2022 4:00:00 PM
1/2/2022 12:00:00 AM
1/2/2022 8:00:00 AM
1/2/2022 4:00:00 PM
1/3/2022 12:00:00 AM
1/3/2022 8:00:00 AM
1/3/2022 4:00:00 PM
1/4/2022 12:00:00 AM
1/4/2022 8:00:00 AM
1/4/2022 4:00:00 PM
1/5/2022 12:00:00 AM
1/5/2022 8:00:00 AM
1/5/2022 4:00:00 PM
1/6/2022 12:00:00 AM
1/6/2022 8:00:00 AM
1/6/2022 4:00:00 PM
1/7/2022 12:00:00 AM
1/7/2022 8:00:00 AM
1/7/2022 4:00:00 PM
1/8/2022 12:00:00 AM
1/8/2022 8:00:00 AM
1/8/2022 4:00:00 PM
1/9/2022 12:00:00 AM
+ 68 weitere Zeilen

Die anfängliche Funktion `autogenerate` erstellt einen Kalender mit allen Datumswerten im Januar in einer Tabelle namens `tmpCalendar`.

Eine zweite Tabelle mit drei Datensätzen wird erstellt. Für jeden Datensatz wird `recno() - 1` mit den Werten 0, 1 und 2 verwendet, und das Ergebnis wird mit 8 multipliziert. Als Ergebnis werden die Werte 0, 8 und 16 generiert. Diese Werte werden als Stundenparameter in einer Funktion `maketime()` verwendet. Die Minuten- und Sekundenwerte sind 0. Als Ergebnis enthält die Tabelle drei Uhrzeitfelder: 12:00:00 AM, 8:00:00 AM und 4:00:00 PM.

Diese Tabelle wird mit der Tabelle `tmpCalendar` verknüpft. Da keine übereinstimmenden Felder zwischen den beiden Feldern für den Join vorhanden sind, werden die Uhrzeitzeilen zu jeder Datumszeile hinzugefügt. Als Ergebnis wird jede Datumszeile für die einzelnen Uhrzeitwerte dreimal wiederholt.

Abschließend wird die Kalendertabelle über einen Resident Load der Tabelle `tmpCalendar` erstellt. Die Datums- und Uhrzeitfelder werden verkettet und in der Funktion `timestamp()` eingeschlossen, um das Zeitstempelfeld zu bilden.

Dann wird die Tabelle `tmpCalendar` gelöscht.

makeweekdate

Diese Funktion gibt ein Datum zurück, das sich aus dem Jahr, der Wochennummer und dem Tag der Woche berechnet.

Syntax:

```
MakeWeekDate(weekyear [, week [, weekday [, first_week_day [, broken_weeks [, reference_day]]]]])
```

Rückgabe Datentyp: dual

Die Funktion `makeweekdate()` ist sowohl als Skript als auch als Diagrammfunktion verfügbar. Mit der Funktion wird das Datum basierend auf den an die Funktion übergebenen Parametern berechnet.

Argumente

Argument	Beschreibung
weekyear	Das Jahr, wie es durch die Funktion `WeekYear()` für das spezifische Datum definiert wird, also das Jahr, zu dem die Wochennummer gehört. *Das Wochenjahr kann sich in einigen Fällen vom Kalenderjahr unterscheiden, zum Beispiel, wenn Woche 1 bereits im Dezember des Vorjahres beginnt.*
week	Die Wochennummer, wie sie durch die Funktion `week()` für das spezifische Datum definiert wird. Wenn keine Wochennummer angegeben ist, wird 1 angenommen.

Argument	Beschreibung
weekday	Der Tag der Woche, wie durch die Funktion `weekDay()` für das betreffende Datum definiert. 0 ist der erste Tag der Woche, 6 der letzte Tag der Woche. Wenn kein Wochentag angegeben ist, wird 0 angenommen. *0 bezieht sich immer auf den ersten Tag der Woche und 6 auf den letzten; welcher Wochentag jedoch der Zahl entspricht, wird durch den Parameter **first_week_day** bestimmt. Ist nichts definiert, wird der Wert der Variablen **FirstWeekDay** verwendet.* Wenn gestückelte Wochen zusammen mit einer unmöglichen Parameterkombination verwendet werden, kann dies zu einem Ergebnis führen, das nicht zum gewählten Jahr gehört. **Beispiel:** `MakeWeekDate(2021,1,0,6,1)` Gibt „Dec 27 2020“ zurück, da dieser Tag der erste Tag (der Sonntag) der angegebenen Woche ist. „Jan 1 2021“ war ein Freitag.
first_week_day	Legt den Tag fest, an dem die Woche beginnt. Ist nichts definiert, wird der Wert der Variable **FirstWeekDay** verwendet. Die möglichen Werte für **first_week_day** sind 0 für Montag, 1 für Dienstag, 2 für Mittwoch, 3 für Donnerstag, 4 für Freitag, 5 für Samstag und 6 für Sonntag. Weitere Informationen über die Systemvariable finden Sie unter *FirstWeekDay (page 233)*.
broken_weeks	Wenn Sie **broken_weeks** nicht angeben, wird der Wert der Variablen **BrokenWeeks** verwendet, um festzulegen, ob die Wochen gestückelt sind oder nicht.
reference_day	Wenn Sie **reference_day** nicht angeben, wird der Wert der Variablen **ReferenceDay** verwendet, um festzulegen, welcher Tag im Januar als Referenztag für die Definition von Woche 1 konfiguriert wird.

Verwendung

Die Funktion `makeweekdate()` wird in der Regel im Skript für die Datengenerierung verwendet, um eine Liste von Datumsangaben zu generieren oder um Datumswerte zu konstruieren, wenn Jahr, Woche und Tag der Woche in den Eingabedaten bereitgestellt werden.

Die folgenden Beispiele nehmen an:

```
SET FirstWeekDay=0;
SET BrokenWeeks=0;
SET ReferenceDay=4;
```

Funktionsbeispiele

Beispiel	Ergebnis
`makeweekdate(2014,6,6)`	gibt 02/09/2014 zurück
`makeweekdate(2014,6,1)`	gibt 02/04/2014 zurück
`makeweekdate(2014,6)`	gibt 02/03/2014 zurück (Wochentag 0 wird angenommen)

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Beispiel 1 – Tag eingeschlossen

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Datensatz mit wöchentlichen Umsätzen für 2022 in einer Tabelle namens `Sales`.
- Transaktionsdatumswerte, die in drei Feldern bereitgestellt werden: `year`, `week` und `sales`.
- Ein vorangehender load-Befehl, der verwendet wird, um eine Kennzahl `end_of_week` zu erstellen. Dabei wird die Funktion `makeweekdate()` verwendet, um das Datum für den Freitag dieser Woche im Format MM/TT/JJJJ zurückzugeben.

Um zu beweisen, dass das zurückgegebene Datum ein Freitag ist, ist die Formel `end_of_week` auch in die Funktion `weekday()` eingeschlossen, um den Tag der Woche zu zeigen.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';
SET FirstWeekDay=0;
SET BrokenWeeks=0;
```

```
SET ReferenceDay=4;

Transactions:
    Load
        *,
        makeweekdate(transaction_year, transaction_week,4) as end_of_week,
        weekday(makeweekdate(transaction_year, transaction_week,4)) as week_day
        ;
Load * Inline [
transaction_year, transaction_week, sales
2022, 01, 10000
2022, 02, 11250
2022, 03, 9830
2022, 04, 14010
2022, 05, 28402
2022, 06, 9992
2022, 07, 7292
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `transaction_year`
- `transaction_week`
- `end_of_week`
- `week_day`

Ergebnistabelle

transaction_year	transaction_week	end_of_week	week_day
2022	01	01/07/2022	Fr
2022	02	01/14/2022	Fr
2022	03	01/21/2022	Fr
2022	04	01/28/2022	Fr
2022	05	02/04/2022	Fr
2022	06	02/11/2022	Fr
2022	07	02/18/2022	Fr

Das Feld `end_of_week` wird im vorangehenden load-Befehl mithilfe der Funktion `makeweekdate()` erstellt. Die Felder `transaction_year` und `transaction_week` werden über die Funktion als die Argumente für Jahr und Woche übergeben. Ein Wert von 4 wird als Argument für Tag verwendet.

Die Funktion kombiniert diese Werte dann und konvertiert sie in ein Datumsfeld. Die Ergebnisse werden im Format der Systemvariablen `DateFormat` zurückgegeben.

Die Funktion makeweekdate() und ihre Argumente sind auch in eine Funktion weekday() eingeschlossen, um das Feld week_day zurückzugeben. Wie in der Tabelle oben zu sehen ist, zeigt das Feld week_day, dass diese Datumswerte tatsächlich an einem Freitag liegen.

Beispiel 2 – Tag ausgeschlossen

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Datensatz mit wöchentlichen Umsätzen für 2022 in einer Tabelle namens Sales
- Transaktionsdatumswerte, die in drei Feldern bereitgestellt werden: year, week und sales.
- Eine vorangehende load-Anweisung, die verwendet wird, um eine Kennzahl first_day_of_week zu erstellen; dabei wird die Funktion makeweekdate() verwendet. Damit wird das Datum für den Montag dieser Woche im Format MM/TT/JJJJ zurückgegeben.

Um zu beweisen, dass das zurückgegebene Datum ein Montag ist, ist die Formel first_day_of_week auch in die Funktion weekday() eingeschlossen, um den Tag der Woche zu zeigen.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';
SET FirstWeekDay=0;
SET BrokenWeeks=0;
SET ReferenceDay=4;

Transactions:
    Load
        *,
        makeweekdate(transaction_year, transaction_week) as first_day_of_week,
        weekday(makeweekdate(transaction_year, transaction_week)) as week_day
        ;
Load * Inline [
transaction_year, transaction_week, sales
2022, 01, 10000
2022, 02, 11250
2022, 03, 9830
2022, 04, 14010
2022, 05, 28402
2022, 06, 9992
2022, 07, 7292
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `transaction_year`
- `transaction_week`
- `first_day_of_week`
- `week_day`

Ergebnistabelle

transaction_year	transaction_week	first_day_of_week	week_day
2022	01	01/03/2022	Mo
2022	02	01/10/2022	Mo
2022	03	01/17/2022	Mo
2022	04	01/24/2022	Mo
2022	05	01/31/2022	Mo
2022	06	02/07/2022	Mo
2022	07	02/14/2022	Mo

Das Feld `first_day_of_week` wird in der vorangehenden load-Anweisung mithilfe der Funktion `makeweekdate()` erstellt. Die Parameter `transaction_year` und `transaction_week` werden als Funktionsargumente übergeben, und der Parameter für Tag wird leer gelassen.

Die Funktion kombiniert diese Werte dann und konvertiert sie in ein Datumsfeld. Die Ergebnisse werden im Format der Systemvariablen `DateFormat` zurückgegeben.

Die Funktion `makeweekdate()` und ihre Argumente sind auch in eine Funktion `weekday()` eingeschlossen, um das Feld `week_day` zurückzugeben. Wie in der Tabelle oben ersichtlich, gibt das Feld `week_day` in allen Fällen Montag zurück, da der Parameter in der Funktion `makeweekdate()` leergelassen wurde. Somit wird standardmäßig `0` (erster Tag der Woche) verwendet, und der erste Tag der Woche ist von der Systemvariablen `FirstWeekDay` auf Montag festgelegt.

Beispiel 3 – Diagrammobjektbeispiel

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Datensatz mit wöchentlichen Umsätzen für 2022 in einer Tabelle namens `Sales`
- Transaktionsdatumswerte, die in drei Feldern bereitgestellt werden: `year`, `week` und `sales`.

In diesem Beispiel wird ein Diagrammobjekt verwendet, um eine Kennzahl zu erstellen, die der Berechnung für `end_of_week` im ersten Beispiel entspricht. Diese Kennzahl verwendet die Funktion `makeweekdate()`, um das Datum für den Freitag dieser Woche im Format MM/TT/JJJJ zurückzugeben.

Um zu beweisen, dass das zurückgegebene Datum an einem Freitag liegt, wird eine zweite Kennzahl erstellt, um den Tag der Woche zurückzugeben.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';
SET FirstWeekDay=0;
SET BrokenWeeks=0;
SET ReferenceDay=4;

Master_Calendar:
Load * Inline [
transaction_year, transaction_week, sales
2022, 01, 10000
2022, 02, 11250
2022, 03, 9830
2022, 04, 14010
2022, 05, 28402
2022, 06, 9992
2022, 07, 7292
];
```

Ergebnisse

Gehen Sie folgendermaßen vor:

1. Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:
   - `transaction_year`
   - `transaction_week`
2. Um die Berechnung durchzuführen, die der des Felds `end_of_week` aus dem ersten Beispiel entspricht, erstellen Sie die folgende Kennzahl:
   `=makeweekdate(transaction_year,transaction_week,4)`
3. Um den Tag der Woche für jede Transaktion zu berechnen, erstellen Sie die folgende Kennzahl:
   `=weekday(makeweekdate(transaction_year,transaction_week,4))`

Ergebnistabelle

transaction_ year	**transaction_ week**	**=makeweekdate (transaction_ year,transaction_week,4)**	**=weekday(makeweekdate (transaction_year,transaction_ week,4))**
2022	01	01/07/2022	Fr

transaction_ year	transaction_ week	=makeweekdate (transaction_ year,transaction_week,4)	=weekday(makeweekdate (transaction_year,transaction_ week,4))
2022	02	01/14/2022	Fr
2022	03	01/21/2022	Fr
2022	04	01/28/2022	Fr
2022	05	02/04/2022	Fr
2022	06	02/11/2022	Fr
2022	07	02/18/2022	Fr

Ein dem `end_of_week` entsprechendes Feld wird im Diagrammobjekt als Kennzahl erstellt, indem die Funktion `makeweekdate()` verwendet wird. Die Felder `transaction_year` und `transaction_week` werden als die Argumente für Jahr und Woche übergeben. Ein Wert von 4 wird als Argument für Tag verwendet.

Die Funktion kombiniert diese Werte dann und konvertiert sie in ein Datumsfeld. Die Ergebnisse werden im Format der Systemvariablen `DateFormat` zurückgegeben.

Die Funktion `makeweekdate()` und ihre Argumente sind auch in eine Funktion `weekday()` eingeschlossen, um eine Berechnung zurückzugeben, die derjenigen für das Feld `week_day` im ersten Beispiel entspricht. Wie in der Tabelle oben zu sehen ist, zeigt die letzte Spalte rechts, dass diese Datumswerte tatsächlich auf einen Freitag fallen.

Beispiel 4 – Szenario

Ladeskript und Diagrammformel

Übersicht

In diesem Beispiel erstellen Sie eine Liste der Datumswerte, die alle Freitage für das Jahr 2022 umfasst.

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';
SET FirstWeekDay=0;
SET BrokenWeeks=0;
SET ReferenceDay=4;

Calendar:
    load
           *,
        weekday(date) as weekday
    where year(date)=2022;
load
    makeweekdate(2022,recno()-2,4) as date
AutoGenerate 60;
```

Ergebnisse

Ergebnistabelle

date	weekday
01/07/2022	Fr
01/14/2022	Fr
01/21/2022	Fr
01/28/2022	Fr
02/04/2022	Fr
02/11/2022	Fr
02/18/2022	Fr
02/25/2022	Fr
03/04/2022	Fr
03/11/2022	Fr
03/18/2022	Fr
03/25/2022	Fr
04/01/2022	Fr
04/08/2022	Fr
04/15/2022	Fr
04/22/2022	Fr
04/29/2022	Fr
05/06/2022	Fr
05/13/2022	Fr
05/20/2022	Fr
05/27/2022	Fr
06/03/2022	Fr
06/10/2022	Fr
06/17/2022	Fr
+ 27 weitere Zeilen	

Die Funktion `makeweekdate()` findet jeden Freitag im Jahr 2022. Ein Wochenparameter von -2 sorgt dafür, dass keine Datumswerte ausgelassen werden. Abschließend erstellt eine vorangehende load-Anweisung ein weiteres Feld `weekday` zur Klarstellung, um zu zeigen, dass jeder Wert für `date` ein Freitag ist.

minute

Diese Funktion liefert die Minute als ganze Zahl, wenn **expression** entsprechend dem Standardformat als Uhrzeit interpretiert wird.

Syntax:

```
minute(expression)
```

Rückgabe Datentyp: ganze Zahl

Verwendung

Die Funktion `minute()` ist nützlich, wenn Sie Aggregierungen nach Minute vergleichen möchten. Sie können die Funktion beispielsweise verwenden, wenn Sie die Verteilung der Anzahl der Aktivitäten nach Minute anzeigen möchten.

Diese Dimensionen können im Ladeskript erstellt werden, indem die Funktion verwendet wird, um ein Feld in einer Master-Kalender-Tabelle zu erstellen. Alternativ können sie direkt als berechnete Dimension in einem Diagramm verwendet werden.

Funktionsbeispiele

Beispiel	Ergebnis
`minute ( '09:14:36' )`	Gibt 14 zurück.
`minute ( '0.5555' )`	Gibt 19 zurück (da 0,5555 = 13:19:55).

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Beispiel 1 – Variable (Skript)

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz, der Transaktionen nach Zeitstempel enthält, wird in eine Tabelle namens `Transactions` geladen.
- Die Standardsystemvariable `TimeStamp` (`M/D/YYYY h:mm:ss[.fff] TT`) wird verwendet.
- Erstellung eines Felds `minute`, um zu berechnen, wann Transaktionen stattfanden

Ladeskript

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Transactions:
    Load
        *,
        minute(timestamp) as minute
        ;
Load
*
Inline
[
id,timestamp,amount
9497,'2022-01-05 19:04:57',47.25,
9498,'2022-01-03 14:21:53',51.75,
9499,'2022-01-03 05:40:49',73.53,
9500,'2022-01-04 18:49:38',15.35,
9501,'2022-01-01 22:10:22',31.43,
9502,'2022-01-05 19:34:46',13.24,
9503,'2022-01-04 22:58:34',74.34,
9504,'2022-01-06 11:29:38',50.00,
9505,'2022-01-02 08:35:54',36.34,
9506,'2022-01-06 08:49:09',74.23
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `timestamp`
- `minute`

Ergebnistabelle

Zeitstempel	minute
2022-01-01 22:10:22	10
2022-01-02 08:35:54	35
2022-01-03 05:40:49	40
2022-01-03 14:21:53	21
2022-01-04 18:49:38	49
2022-01-04 22:58:34	58
2022-01-05 19:04:57	4
2022-01-05 19:34:46	34
2022-01-06 08:49:09	49
2022-01-06 11:29:38	29

Die Werte im Feld `minute` werden anhand der Funktion `minute()` erstellt und übergeben den `timestamp` als die Formel im vorangehenden load-Befehl.

Beispiel 2 – Diagrammobjekt (Diagramm)

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Derselbe Datensatz und dasselbe Szenario wie im ersten Beispiel.
- Die Standardsystemvariable `TimeStamp` (`M/D/YYYY h:mm:ss[.fff] TT`) wird verwendet.

In diesem Beispiel wird jedoch der unveränderte Datensatz in die Anwendung geladen. Die Werte für `minute` werden anhand einer Kennzahl in einem Diagrammobjekt berechnet.

Ladeskript

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Transactions:
Load
*
Inline
[
id,timestamp,amount
9497,'2022-01-05 19:04:57',47.25,
9498,'2022-01-03 14:21:53',51.75,
9499,'2022-01-03 05:40:49',73.53,
```

```
9500,'2022-01-04 18:49:38',15.35,
9501,'2022-01-01 22:10:22',31.43,
9502,'2022-01-05 19:34:46',13.24,
9503,'2022-01-04 22:58:34',74.34,
9504,'2022-01-06 11:29:38',50.00,
9505,'2022-01-02 08:35:54',36.34,
9506,'2022-01-06 08:49:09',74.23
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu: `timestamp`.

Erstellen Sie die folgende Kennzahl:

```
=minute(timestamp)
```

Ergebnistabelle

Zeitstempel	minute
2022-01-01 22:10:22	10
2022-01-02 08:35:54	35
2022-01-03 05:40:49	40
2022-01-03 14:21:53	21
2022-01-04 18:49:38	49
2022-01-04 22:58:34	58
2022-01-05 19:04:57	4
2022-01-05 19:34:46	34
2022-01-06 08:49:09	49
2022-01-06 11:29:38	29

Die Werte für `minute` werden erstellt, indem die Funktion `minute()` verwendet und der `timestamp` als Formel in einer Kennzahl für das Diagrammobjekt übergeben wird.

Beispiel 3 – Szenario

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz mit Zeitstempeln wird generiert, um eingelassene Personen an einer Sperre darzustellen.
- Informationen für jeden `timestamp` und die zugehörige `id` werden in eine Tabelle namens `Ticket_Barrier_Tracker` geladen.
- Die Standardsystemvariable `TimeStamp` (`M/D/YYYY h:mm:ss[.fff] TT`) wird verwendet.

Der Benutzer möchte ein Diagrammobjekt, das nach Minute die Anzahl der an der Sperre eingelassenen Personen anzeigt.

Ladeskript

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

tmpTimeStampCreator:
       load
               *
    where year(date)=2022;
load
       date(recno()+makedate(2021,12,31)) as date
AutoGenerate 1;

join load
       maketime(floor(rand()*24),floor(rand()*59),floor(rand()*59)) as time
autogenerate 10000;

Ticket_Barrier_Tracker:
load
       recno() as id,
       timestamp(date + time) as timestamp
resident tmpTimeStampCreator;

drop table tmpTimeStampCreator;
```

Ergebnisse

Gehen Sie folgendermaßen vor:

1. Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle.
2. Erstellen Sie eine berechnete Dimension anhand der folgenden Formel:
   `=minute(timestamp)`
3. Fügen Sie die folgende Aggregierungskennzahl hinzu, um die Gesamtzahl der eingelassenen Personen zu berechnen:
   `=count(id)`
4. Legen Sie das **Zahlenformat** der Kennzahl auf **Währung** fest.

Ergebnistabelle

minute(timestamp)	=count(id)
0	174
1	171

minute(timestamp)	=count(id)
2	175
3	165
4	188
5	176
6	158
7	187
8	178
9	178
10	197
11	161
12	166
13	184
14	159
15	161
16	152
17	160
18	176
19	164
20	170
21	170
22	142
23	145
24	155
+ 35 weitere Zeilen	

month

Diese Funktion gibt einen dualen Wert zurück: ein Monatsname gemäß Definition in der Umgebungvariable **MonthNames** sowie eine Ganzzahl zwischen 1-12. Der Monat berechnet sich durch die Datumsinterpretation der Formel entsprechend dem Standardformat.

Die Funktion gibt den Namen des Monats im Format der Systemvariablen `MonthName` für ein bestimmtes Datum zurück. Sie wird gewöhnlich verwendet, um ein Tagesfeld als Dimension in einem Master-Kalender zu erstellen.

Syntax:

```
month(expression)
```

Rückgabe Datentyp: ganze Zahl

Funktionsbeispiele

Beispiel	Ergebnis
month(2012-10-12)	liefert Oct
month(35648)	liefert Aug, da 35648 = 1997-08-06

Beispiel 1 – DateFormat-Datensatz (Skript)

Ladeskript und Ergebnisse

Überblick

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Einen Datensatz mit Datumsangaben mit dem Namen `Master_Calendar`. Die Systemvariable `DateFormat` wird auf TT/MM/JJJJ festgelegt.
- Einen vorangehenden load-Befehl, mit dem unter Verwendung der Funktion `month()` ein zusätzliches Feld mit dem Namen `month_name` erstellt wird.
- Ein weiteres Feld mit dem Namen `long_date`, das die Funktion `date()` verwendet, um das vollständige Datum auszudrücken.

Ladeskript

```
SET DateFormat='DD/MM/YYYY';

Master_Calendar:
Load
        date,
        date(date,'dd-MMMM-YYYY') as long_date,
        month(date) as month_name
Inline
[
date
03/01/2022
03/02/2022
03/03/2022
03/04/2022
03/05/2022
03/06/2022
03/07/2022
03/08/2022
```

```
03/09/2022
03/10/2022
03/11/2022
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `long_date`
- `month_name`

Ergebnistabelle

date	long_date	month_name
03/01/2022	03-January- 2022	Jan
03/02/2022	03-February- 2022	Feb
03/03/2022	03-March- 2022	Mär
03/04/2022	03-April- 2022	Apr
03/05/2022	03-May- 2022	Mai
03/06/2022	03-June- 2022	Jun
03/07/2022	03-July- 2022	Jul
03/08/2022	03-August- 2022	Aug
03/09/2022	03-September- 2022	Sep
03/10/2022	03-October- 2022	Oct
03/11/2022	03-November- 2022	Nov

Der Name des Monats wird korrekt von der Funktion `month()` im Skript ausgewertet.

Beispiel 2 – ANSI-Datum (Skript)

Ladeskript und Ergebnisse

Überblick

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Einen Datensatz mit Datumsangaben mit dem Namen `Master_Calendar`. Die DateFormat-Systemvariable TT/MM/JJJJ wird verwendet. Die im Datensatz enthaltenen Datumsangaben weisen aber das ANSI-Standarddatumsformat auf.

- Einen vorangehenden load-Befehl, mit dem ein zusätzliches Feld mit dem Namen `month_name` erstellt wird, unter Verwendung der Funktion month().
- Ein weiteres Feld mit dem Namen `long_date`, das die Funktion date() verwendet, um das vollständige Datum auszudrücken.

Ladeskript

```
SET DateFormat='DD/MM/YYYY';
Master_Calendar:
Load
       date,
       date(date,'dd-MMMM-YYYY') as long_date,
       month(date) as month_name

Inline
[
date
2022-01-11
2022-02-12
2022-03-13
2022-04-14
2022-05-15
2022-06-16
2022-07-17
2022-08-18
2022-09-19
2022-10-20
2022-11-21
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `long_date`
- `month_name`

Ergebnistabelle

date	long_date	month_name
03/11/2022	11-March- 2022	11
03/12/2022	12-March- 2022	12
03/13/2022	13-March- 2022	13
03/14/2022	14-March- 2022	14
03/15/2022	15-March- 2022	15

date	long_date	month_name
03/16/2022	16-March- 2022	16
03/17/2022	17-March- 2022	17
03/18/2022	18-March- 2022	18
03/19/2022	19-March- 2022	19
03/20/2022	20-March- 2022	20
03/21/2022	21-March- 2022	21

Der Name des Monats wird korrekt von der Funktion `month()` im Skript ausgewertet.

Beispiel 3 – Unformatiertes Datum (Skript)

Ladeskript und Ergebnisse

Überblick

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Einen Datensatz mit Datumsangaben mit dem Namen `Master_Calendar`. Die `DateFormat`-Systemvariable TT/MM/JJJJ wird verwendet.
- Einen vorangehenden load-Befehl, mit dem ein zusätzliches Feld mit dem Namen `month_name` erstellt wird, unter Verwendung der Funktion `month()` .
- Das ursprüngliche unformatierte Datum mit dem Namen `unformatted_date`.
- Ein weiteres Feld mit dem Namen `long_date`, das die Funktion `date()` verwendet, um das vollständige Datum auszudrücken.

Ladeskript

```
SET DateFormat='DD/MM/YYYY';

Master_Calendar:
Load
      unformatted_date,
      date(unformatted_date,'dd-MMMM-YYYY') as long_date,
      month(unformatted_date) as month_name

Inline
[
unformatted_date
44868
44898
44928
44958
44988
45018
```

```
45048
45078
45008
45038
45068
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `unformatted_date`
- `long_date`
- `month_name`

Ergebnistabelle

unformatted_date	long_date	month_name
44868	03-January- 2022	Jan
44898	03-February- 2022	Feb
44928	03-March- 2022	Mär
44958	03-April- 2022	Apr
44988	03-May- 2022	Mai
45018	03-June- 2022	Jun
45048	03-July- 2022	Jul
45078	03-August- 2022	Aug
45008	03-September- 2022	Sep
45038	03-October- 2022	Oct
45068	03-November- 2022	Nov

Der Name des Monats wird korrekt von der Funktion `month()` im Skript ausgewertet.

Beispiel 4 – Berechnen des Ablaufmonats

Ladeskript und Diagrammformel

Überblick

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz der im März aufgegebenen Bestellungen mit dem Namen `Subscriptions`. Die Tabelle enthält drei Felder:
    - id
    - order_date
    - amount

Ladeskript

```
Subscriptions:
Load
        id,
        order_date,
        amount
Inline
[
id,order_date,amount
1,03/01/2022,231.24
2,03/02/2022,567.28
3,03/03/2022,364.28
4,03/04/2022,575.76
5,03/05/2022,638.68
6,03/06/2022,785.38
7,03/07/2022,967.46
8,03/08/2022,287.67
9,03/09/2022,764.45
10,03/10/2022,875.43
11,03/11/2022,957.35
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu: `order_date`.

Zum Berechnen des Monats, in dem ein Auftrag abläuft, erstellen Sie die folgende Kennzahl: `=month(order_date+180)`.

Ergebnistabelle

order_date	=month(order_date+180)
03/01/2022	Jul
03/02/2022	Aug
03/03/2022	Aug
03/04/2022	Sep
03/05/2022	Oct
03/06/2022	Nov

order_date	=month(order_date+180)
03/07/2022	Dec
03/08/2022	Jan
03/09/2022	Mär
03/10/2022	Apr
03/11/2022	Mai

Die Funktion `month()` berechnet korrekt, dass eine am 11. März aufgegebene Bestellung im Juli abläuft.

monthend

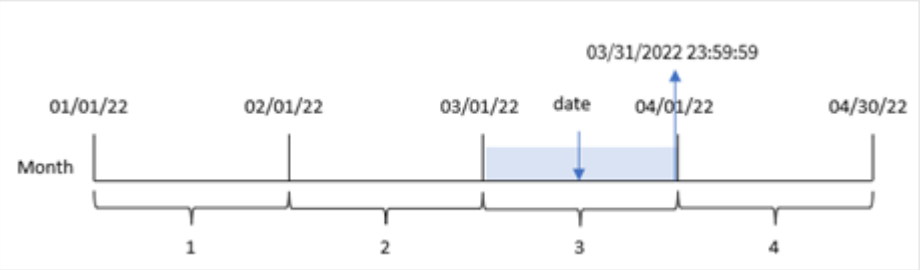
Diese Funktion liefert den Zeitstempel der letzten Millisekunde des letzten Tags des Monats, in dem `date` liegt. Das Ergebnis wird entsprechend dem im Skript definierten `DateFormat` formatiert.

Syntax:

```
MonthEnd(date[, period_no])
```

Die Funktion `monthend()` legt also fest, in welchen Monat das Datum fällt. Sie gibt dann einen Zeitstempel im Datumsformat für die letzte Millisekunde dieses Monats zurück.

Diagramm der Funktion monthend.



Verwendung

Die Funktion `monthend()` wird als Teil einer Formel verwendet, wenn in der Berechnung der Teil des Monats verwendet werden soll, der noch nicht eingetreten ist. Beispiel: Sie möchten die gesamten, während des Monats noch nicht fällig gewordenen Zinsen berechnen.

Rückgabe Datentyp: dual

Argumente

Argument	Beschreibung
date	Datum oder Zeitstempel für die Evaluierung.
period_no	**period_no** ist eine ganze Zahl, die beim Weglassen von 0 den Monat liefert, der **date** enthält. Negative Werte von **period_no** stehen für vorangehende, positive Werte für nachfolgende Monate.

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Funktionsbeispiele

Beispiel	**Ergebnis**
`monthend('02/19/2012')`	Gibt 02/29/2012 23:59:59 zurück.
`monthend('02/19/2001', -1)`	Gibt 01/31/2001 23:59:59 zurück.

Beispiel 1 – einfaches Beispiel

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Datensatz mit einer Reihe von Transaktionen für 2022, der in eine Tabelle namens „`Transactions`“ geladen wird
- Datumsfeld im Format `MM/DD/YYYY` der Systemvariablen `DateFormat`.
- Ein vorangehender load-Befehl, der Folgendes enthält:
    - Die Funktion `monthend()`, die als Feld „`end_of_month`“ festgelegt ist.
    - Die Funktion `timestamp`, die als Feld „`end_of_month_timestamp`“ festgelegt ist.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
    *,
    monthend(date) as end_of_month,
    timestamp(monthend(date)) as end_of_month_timestamp
```

```
    ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `id`
- `date`
- `end_of_month`
- `end_of_month_timestamp`

Ergebnistabelle

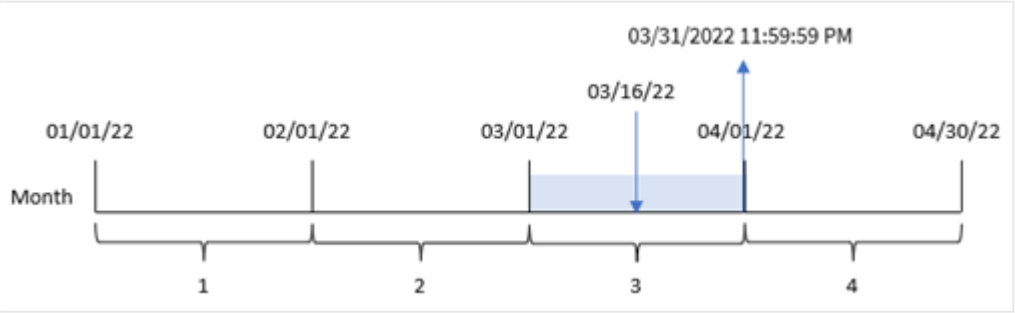
id	date	end_of_month	end_of_month_timestamp
8188	1/7/2022	01/31/2022	1/31/2022 11:59:59 PM
8189	1/19/2022	01/31/2022	1/31/2022 11:59:59 PM
8190	2/5/2022	02/28/2022	2/28/2022 11:59:59 PM
8191	2/28/2022	02/28/2022	2/28/2022 11:59:59 PM
8192	3/16/2022	03/31/2022	3/31/2022 11:59:59 PM
8193	4/1/2022	04/30/2022	4/30/2022 11:59:59 PM

id	date	end_of_month	end_of_month_timestamp
8194	5/7/2022	05/31/2022	5/31/2022 11:59:59 PM
8195	5/16/2022	05/31/2022	5/31/2022 11:59:59 PM
8196	6/15/2022	06/30/2022	6/30/2022 11:59:59 PM
8197	6/26/2022	06/30/2022	6/30/2022 11:59:59 PM
8198	7/9/2022	07/31/2022	7/31/2022 11:59:59 PM
8199	7/22/2022	07/31/2022	7/31/2022 11:59:59 PM
8200	7/23/2022	07/31/2022	7/31/2022 11:59:59 PM
8201	7/27/2022	07/31/2022	7/31/2022 11:59:59 PM
8202	8/2/2022	08/31/2022	8/31/2022 11:59:59 PM
8203	8/8/2022	08/31/2022	8/31/2022 11:59:59 PM
8204	8/19/2022	08/31/2022	8/31/2022 11:59:59 PM
8205	9/26/2022	09/30/2022	9/30/2022 11:59:59 PM
8206	10/14/2022	10/31/2022	10/31/2022 11:59:59 PM
8207	10/29/2022	10/31/2022	10/31/2022 11:59:59 PM

Das Feld „end_of_month“ wird in der vorangehenden load-Anweisung erstellt, indem die Funktion `monthend()` verwendet und das Datumsfeld als Argument der Funktion übergeben wird.

Die Funktion `monthend()` identifiziert, in welchen Monat der Datumswert fällt, und gibt einen Zeitstempel für die letzte Millisekunde dieses Monats zurück.

*Diagramm der Funktion **monthend** mit März als ausgewähltem Monat.*



Transaktion 8192 fand am 16. März statt. Die Funktion `monthend()` gibt die letzte Millisekunde dieses Monats zurück, also den 31. März um 11:59:59 PM.

Beispiel 2 – period_no

Ladeskript und Ergebnisse

Übersicht

Es werden derselbe Datensatz und dasselbe Szenario wie im ersten Beispiel verwendet.

In diesem Beispiel besteht die Aufgabe darin, ein Feld „previous_month_end“ zu erstellen, das den Zeitstempel für das Ende des Monats vor dem Transaktionsdatum zurückgibt.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
    *,
    monthend(date,-1) as previous_month_end,
    timestamp(monthend(date,-1)) as previous_month_end_timestamp
    ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- id
- date

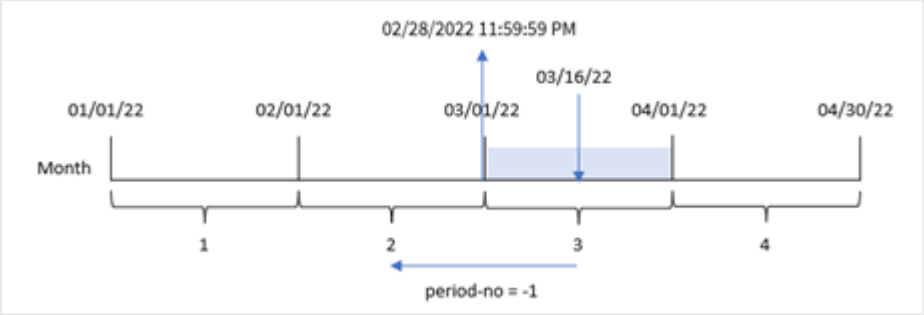
- previous_month_end
- previous_month_end_timestamp

Ergebnistabelle

id	date	previous_month_end	previous_month_end_timestamp
8188	1/7/2022	12/31/2021	12/31/2021 11:59:59 PM
8189	1/19/2022	12/31/2021	12/31/2021 11:59:59 PM
8190	2/5/2022	01/31/2022	1/31/2022 11:59:59 PM
8191	2/28/2022	01/31/2022	1/31/2022 11:59:59 PM
8192	3/16/2022	02/28/2022	2/28/2022 11:59:59 PM
8193	4/1/2022	03/31/2022	3/31/2022 11:59:59 PM
8194	5/7/2022	04/30/2022	4/30/2022 11:59:59 PM
8195	5/16/2022	04/30/2022	4/30/2022 11:59:59 PM
8196	6/15/2022	05/31/2022	5/31/2022 11:59:59 PM
8197	6/26/2022	05/31/2022	5/31/2022 11:59:59 PM
8198	7/9/2022	06/30/2022	6/30/2022 11:59:59 PM
8199	7/22/2022	06/30/2022	6/30/2022 11:59:59 PM
8200	7/23/2022	06/30/2022	6/30/2022 11:59:59 PM
8201	7/27/2022	06/30/2022	6/30/2022 11:59:59 PM
8202	8/2/2022	07/31/2022	7/31/2022 11:59:59 PM
8203	8/8/2022	07/31/2022	7/31/2022 11:59:59 PM
8204	8/19/2022	07/31/2022	7/31/2022 11:59:59 PM
8205	9/26/2022	08/31/2022	8/31/2022 11:59:59 PM
8206	10/14/2022	09/30/2022	9/30/2022 11:59:59 PM
8207	10/29/2022	09/30/2022	9/30/2022 11:59:59 PM

Die Funktion `monthend()` identifiziert zuerst den Monat, in dem die Transaktionen stattfanden, da eine `period_no` von `-1` als Versatzargument verwendet wird. Dann geht sie einen Monat zurück und identifiziert die letzte Millisekunde dieses Monats.

*Diagramm der Funktion **monthend** mit der Variablen „period_no“.*



Transaktion 8192 fand am 16. März statt. Die Funktion `monthend()` identifiziert, dass der Monat vor dem Transaktionsdatum der Februar war. Dann wird die letzte Millisekunde dieses Monats zurückgegeben, der 28. Februar um 11:59:59 PM.

Beispiel 3 – Diagrammbeispiel

Ladeskript und Diagrammformel

Übersicht

Es werden derselbe Datensatz und dasselbe Szenario wie im ersten Beispiel verwendet.

In diesem Beispiel wird der unveränderte Datensatz in die App geladen. Die Aufgabe besteht im Erstellen einer Berechnung, die einen Zeitstempel für das Ende des Monats zurückgibt, in dem die Transaktionen stattfanden. Sie wird als Kennzahl in einem Diagrammobjekt der Anwendung erstellt.

Ladeskript

```
Transactions:
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
```

```
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `id`

Um das Enddatum des Monats zu berechnen, in dem eine Transaktion stattfindet, erstellen Sie die folgenden Kennzahlen:

- `=monthend(date)`
- `=timestamp(monthend(date))`

Ergebnistabelle

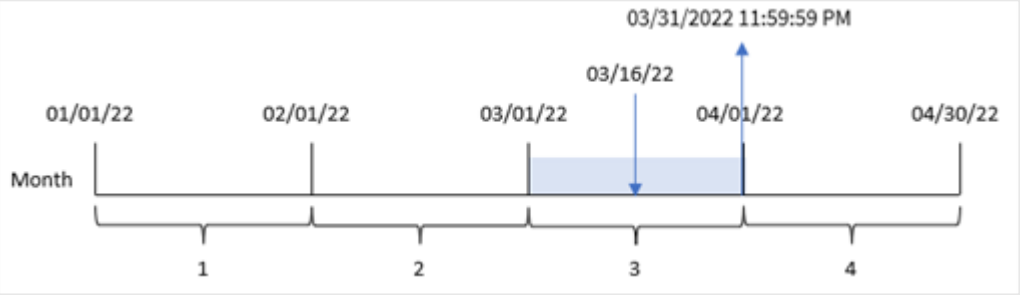
id	date	=monthend(date)	=timestamp(monthend(date))
8188	10/14/2022	10/31/2022	10/31/2022 11:59:59 PM
8189	10/29/2022	10/31/2022	10/31/2022 11:59:59 PM
8190	9/26/2022	09/30/2022	9/30/2022 11:59:59 PM
8191	8/2/2022	08/31/2022	8/31/2022 11:59:59 PM
8192	8/8/2022	08/31/2022	8/31/2022 11:59:59 PM
8193	8/19/2022	08/31/2022	8/31/2022 11:59:59 PM
8194	7/9/2022	07/31/2022	7/31/2022 11:59:59 PM
8195	7/22/2022	07/31/2022	7/31/2022 11:59:59 PM
8196	7/23/2022	07/31/2022	7/31/2022 11:59:59 PM
8197	7/27/2022	07/31/2022	7/31/2022 11:59:59 PM
8198	6/15/2022	06/30/2022	6/30/2022 11:59:59 PM
8199	6/26/2022	06/30/2022	6/30/2022 11:59:59 PM
8200	5/7/2022	05/31/2022	5/31/2022 11:59:59 PM
8201	5/16/2022	05/31/2022	5/31/2022 11:59:59 PM
8202	4/1/2022	04/30/2022	4/30/2022 11:59:59 PM
8203	3/16/2022	03/31/2022	3/31/2022 11:59:59 PM
8204	2/5/2022	02/28/2022	2/28/2022 11:59:59 PM

id	date	=monthend(date)	=timestamp(monthend(date))
8205	2/28/2022	02/28/2022	2/28/2022 11:59:59 PM
8206	1/7/2022	01/31/2022	1/31/2022 11:59:59 PM
8207	1/19/2022	01/31/2022	1/31/2022 11:59:59 PM

Die Kennzahl „end_of_month“ wird im Diagramm erstellt, indem die Funktion `monthend()` verwendet und das Datumsfeld als Argument der Funktion übergeben wird.

Die Funktion `monthend()` identifiziert, in welchen Monat der Datumswert fällt, und gibt einen Zeitstempel für die letzte Millisekunde dieses Monats zurück.

*Diagramm der Funktion **monthend** mit der Variablen „period_no“.*



Transaktion 8192 fand am 16. März statt. Die Funktion `monthend()` gibt die letzte Millisekunde dieses Monats zurück, also den 31. März um 11:59:59 PM.

Beispiel 4 – Szenario

Ladeskript und Ergebnisse

Übersicht

In diesem Beispiel wird ein Datensatz in eine Tabelle namens „Employee_Expenses“ geladen. Die Tabelle enthält die folgenden Felder:

- Mitarbeiter-IDs
- Mitarbeiternamen
- Die durchschnittlichen täglichen Spesenanträge pro Mitarbeiter.

Der Endbenutzer möchte ein Diagramm, das nach Mitarbeiter-ID und Mitarbeiternamen die geschätzten Spesenanträge anzeigt, die für den restlichen Monat noch anfallen.

Ladeskript

```
Employee_Expenses:
Load
*
Inline
[
```

```
employee_id,employee_name,avg_daily_claim
182,Mark, $15
183,Deryck, $12.5
184,Dexter, $12.5
185,Sydney,$27
186,Agatha,$18
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `employee_id`
- `employee_name`

Erstellen Sie die folgende Kennzahl, um die kumulierten Zinsen zu berechnen:

```
=floor(monthend(today(1),0)-today(1))*avg_daily_claim
```

Diese Kennzahl ist dynamisch und ergibt unterschiedliche Tabellenergebnisse, je nach dem Datum, an dem Sie die Daten laden.

Legen Sie das **Zahlenformat** der Kennzahl auf **Währung** fest.

Ergebnistabelle

employee_id	employee_name	=floor(monthend(today(1),0)-today(1))*avg_daily_claim
182	Mark	$30.00
183	Deryck	$25.00
184	Dexter	$25.00
185	Sydney	$54.00
186	Agatha	$36.00

Die Funktion `monthend()` gibt das Enddatum des aktuellen Monats zurück, indem das aktuelle Datum als einziges Argument verwendet wird. Die Formel gibt die Anzahl der im Monat verbleibenden Tage zurück, indem das aktuelle Datum vom Monatsenddatum abgezogen wird.

Dieser Wert wird dann mit den durchschnittlichen täglichen Spesenanträgen der einzelnen Mitarbeitern multipliziert, um den geschätzten Spesenbetrag pro Mitarbeiter für den verbleibende Monat zu berechnen.

monthname

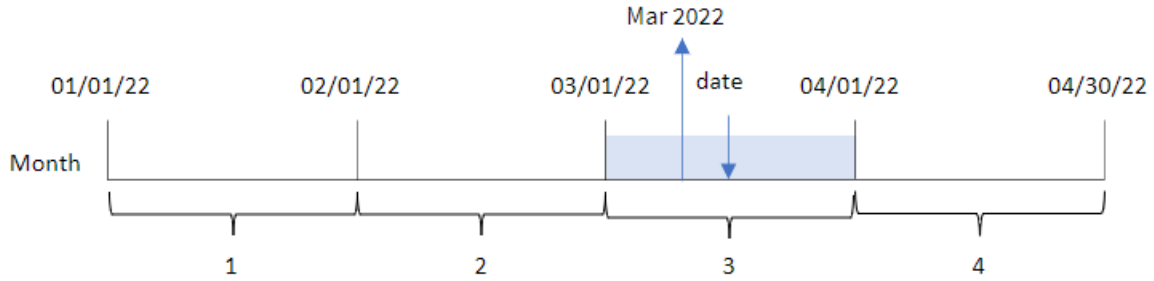
Diese Funktion liefert einen Wert mit dem Monat (entsprechend der Skriptvariable **MonthNames** formatiert) und dem Jahr mit einem numerischen Wert, der dem Zeitstempel der ersten Millisekunde des ersten Tags des Monats entspricht.

Syntax:

```
MonthName(date[, period_no])
```

Rückgabe Datentyp: dual

Diagramm der Funktion „monthname“



Argumente

Argument	Beschreibung
date	Datum oder Zeitstempel für die Evaluierung.
period_no	**period_no** ist eine ganze Zahl, die beim Weglassen von 0 den Monat liefert, der **date** enthält. Negative Werte von **period_no** stehen für vorangehende, positive Werte für nachfolgende Monate.

Funktionsbeispiele

Beispiel	Ergebnis
`monthname('10/19/2013')`	Gibt Oct 2013 zurück
`monthname('10/19/2013', -1)`	Liefert Sep 2013

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Beispiel 1 – einfaches Beispiel

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz enthält eine Reihe von Transaktionen für 2022 und wird in eine Tabelle namens `Transactions` geladen.
- Das Datumsfeld wird im Format der Systemvariablen `DateFormat` (MM/TT/JJJJ) bereitgestellt.
- Die Erstellung eines Felds `transaction_month`, das den Monat zurückgibt, in dem die Transaktionen stattfanden.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';
SET MonthNames='Jan;Feb;Mar;Apr;May;Jun;Jul;Aug;Sep;Oct;Nov;Dec';

Transactions:
    Load
        *,
        monthname(date) as transaction_month
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

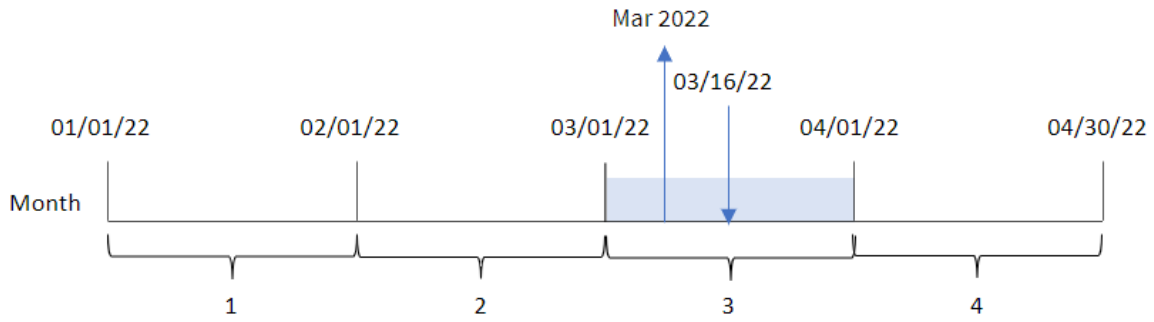
- `date`
- `transaction_month`

Ergebnistabelle

date	transaction_month
1/7/2022	Jan 2022
1/19/2022	Jan 2022
2/5/2022	Feb 2022
2/28/2022	Feb 2022
3/16/2022	Mar 2022
4/1/2022	Apr 2022
5/7/2022	May 2022
5/16/2022	May 2022
6/15/2022	Jun 2022
6/26/2022	Jun 2022
7/9/2022	Jul 2022
7/22/2022	Jul 2022
7/23/2022	Jul 2022
7/27/2022	Jul 2022
8/2/2022	Aug 2022
8/8/2022	Aug 2022
8/19/2022	Aug 2022
9/26/2022	Sep 2022
10/14/2022	Oct 2022
10/29/2022	Oct 2022

Das Feld `transaction_month` wird im vorangehenden load-Befehl erstellt, indem die Funktion `monthname()` verwendet und das Feld `date` als Argument der Funktion übergeben wird.

Diagramm der Funktion „monthname“, einfaches Beispiel



Die Funktion `monthname()` identifiziert, dass die Transaktion 8192 im März 2022 stattfand und gibt diesen Wert anhand der Systemvariablen `MonthNames` zurück.

Beispiel 2 – period_no

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Derselbe Inline-Datensatz und dasselbe Szenario wie im ersten Beispiel.
- Die Erstellung eines Felds `transaction_previous_month`, das den Zeitstempel für das Ende des Monats zurückgibt, bevor die Transaktion stattfand.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';
SET MonthNames='Jan;Feb;Mar;Apr;May;Jun;Jul;Aug;Sep;Oct;Nov;Dec';

Transactions:
    Load
        *,
        monthname(date,-1) as transaction_previous_month
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
```

```
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `transaction_previous_month`

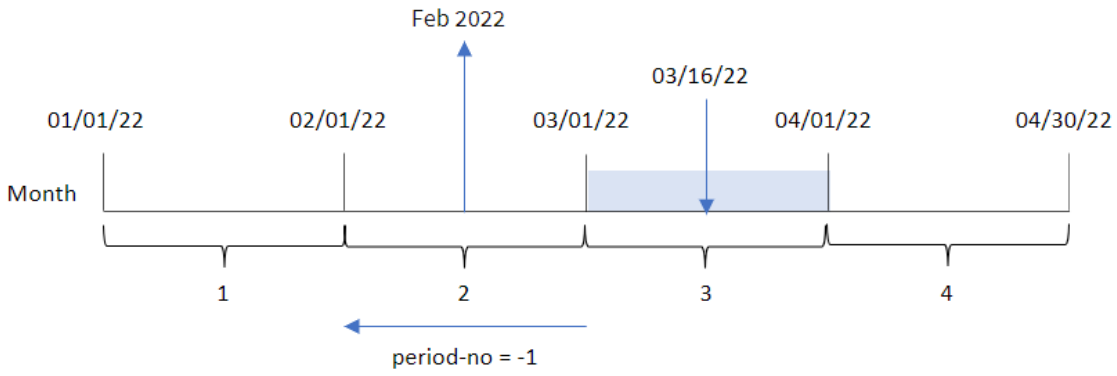
Ergebnistabelle

date	**transaction_previous_month**
1/7/2022	Dec 2021
1/19/2022	Dec 2021
2/5/2022	Jan 2022
2/28/2022	Jan 2022
3/16/2022	Feb 2022
4/1/2022	Mar 2022
5/7/2022	Apr 2022
5/16/2022	Apr 2022
6/15/2022	May 2022
6/26/2022	May 2022
7/9/2022	Jun 2022
7/22/2022	Jun 2022
7/23/2022	Jun 2022
7/27/2022	Jun 2022
8/2/2022	Jul 2022

date	transaction_previous_month
8/8/2022	Jul 2022
8/19/2022	Jul 2022
9/26/2022	Aug 2022
10/14/2022	Sep 2022
10/29/2022	Sep 2022

Da in dieser Instanz eine `period_no` von -1 als Versatzargument in der Funktion `monthname()` verwendet wurde, identifiziert die Funktion zuerst den Monat, in dem die Transaktionen stattfanden. Dann geht sie zu einem Monat vorher und gibt den Monatsnamen und das Jahr zurück.

Diagramm der Funktion „monthname“, Beispiel „period_no“



Transaktion 8192 fand am 16. März statt. Die Funktion `monthname()` identifiziert, dass der Monat, bevor die Transaktion stattfand, Februar war, und gibt den Monat im Format der Systemvariablen `MonthNames` zurück, zusammen mit dem Jahr 2022.

Beispiel 3 – Diagrammobjektbeispiel

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript verwendet den gleichen Inline-Datensatz und das gleiche Szenario wie das erste Beispiel. In diesem Beispiel wird jedoch der unveränderte Datensatz in die Anwendung geladen. Die Berechnung, die einen Zeitstempel für das Ende des Monats zurückgibt, in dem die Transaktionen stattfanden, wird als Kennzahl in einem Diagrammobjekt der Anwendung erstellt.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';
SET MonthNames='Jan;Feb;Mar;Apr;May;Jun;Jul;Aug;Sep;Oct;Nov;Dec';
```

```
Transactions:
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu:`date`.

Erstellen Sie die folgende Kennzahl:

```
=monthname(date)
```

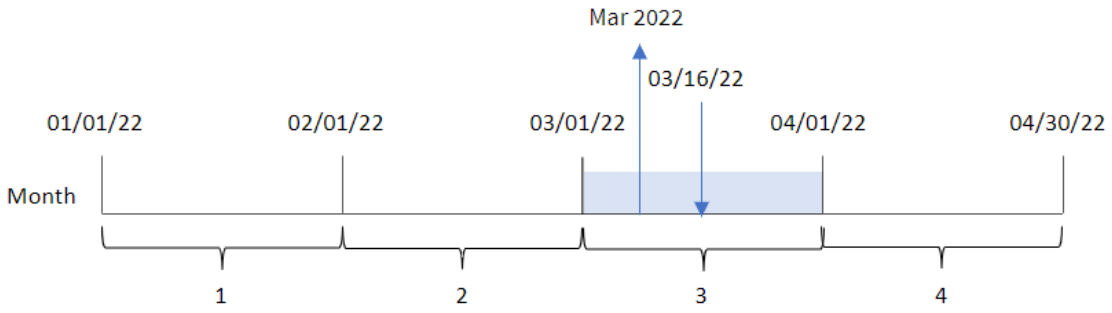
Ergebnistabelle

date	=monthname(date)
1/7/2022	Jan 2022
1/19/2022	Jan 2022
2/5/2022	Feb 2022
2/28/2022	Feb 2022
3/16/2022	Mar 2022
4/1/2022	Apr 2022
5/7/2022	May 2022

date	=monthname(date)
5/16/2022	May 2022
6/15/2022	Jun 2022
6/26/2022	Jun 2022
7/9/2022	Jul 2022
7/22/2022	Jul 2022
7/23/2022	Jul 2022
7/27/2022	Jul 2022
8/2/2022	Aug 2022
8/8/2022	Aug 2022
8/19/2022	Aug 2022
9/26/2022	Sep 2022
10/14/2022	Oct 2022
10/29/2022	Oct 2022

Die Kennzahl `month_name` wird im Diagrammobjekt erstellt, indem die Funktion `monthname()` verwendet und das Feld `date` als Argument der Funktion übergeben wird.

Diagramm der Funktion „monthname“, Diagrammobjektbeispiel



Die Funktion `monthname()` identifiziert, dass die Transaktion 8192 im März 2022 stattfand und gibt diesen Wert anhand der Systemvariablen `MonthNames` zurück.

monthsend

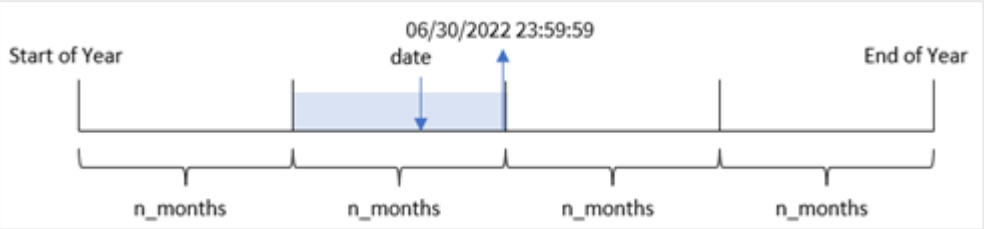
Diese Funktion gibt einen Wert zurück, der einem Zeitstempel für die letzte Millisekunde im Monat, Zweimonatszeitraum, Quartal, Viermonatszeitraum oder Halbjahr entspricht, in dem ein Basisdatum liegt. Es lässt sich auch der Zeitstempel für das Ende eines vorhergehenden oder nachfolgenden Zeitraums bestimmen. Das Ergebnis wird entsprechend dem im Skript definierten `DateFormat` formatiert.

Syntax:

```
MonthsEnd(n_months, date[, period_no [, first_month_of_year]])
```

Rückgabe Datentyp: dual

Diagramm der Funktion monthsend.



Argumente

Argument	Beschreibung
n_months	Die Anzahl der Monate, die den Zeitraum definiert. Eine Ganzzahl oder eine Formel, die eine Ganzzahl mit einem der folgenden Werte ergibt: 1 (entspricht der Funktion inmonth()), 2 (Zweimonatszeitraum), 3 (entspricht der Funktion inquarter()), 4 (Viermonatszeitraum) oder 6 (Halbjahr).
date	Datum oder Zeitstempel für die Evaluierung.
period_no	Mit **period_no**, einer ganze Zahl oder einer Formel, die eine ganze Zahl ergibt, kann ein anderer Beginn für den Zeitraum festgelegt werden, wobei 0 für den Zeitraum steht, der **base_date** enthält. Negative Werte von **period_no** stehen für vorangehende, positive Werte für nachfolgende Zeiträume.
first_ month_of_ year	Wenn Sie mit abweichenden Geschäftsjahren arbeiten möchten, definieren Sie mit einer Zahl zwischen 2 und 12 für **first_month_of_year** einen anderen Startmonat.

Die Funktion `monthsend()` unterteilt das Jahr in Segmente, gestützt auf das angegebene Argument `n_months`. Dann wertet sie aus, in welches Segment jedes angegebene Datum fällt, und gibt die letzte Millisekunde dieses Segments im Datumsformat zurück. Die Funktion kann den Endzeitstempel von vorherigen oder nachfolgenden Segmenten zurückgeben und den ersten Monat des Jahres neu definieren.

Die folgenden Segmente des Jahres sind in der Funktion als Argumente `n_month`verfügbar.

Argumente `n_month`

Zeitraum	Anzahl der Monate
Monat	1
Zweimonatszeitraum	2
Quartal	3
Viermonatszeitraum	4
Halbjahr	6

Verwendung

Die Funktion `monthsend()` wird als Teil einer Formel verwendet, wenn in der Berechnung der Teil des Monats verwendet werden soll, der bereits verstrichen ist. Der Benutzer kann anhand einer Variablen den gewünschten Zeitraum auswählen. Beispielsweise kann `monthsend()` eine Eingabevariable angeben, mit der der Benutzer die gesamten während des Monats, Quartals oder Halbjahres noch nicht angefallenen Zinsen berechnen kann.

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Funktionsbeispiele

Beispiel	**Ergebnis**
`monthsend(4, '07/19/2013')`	Gibt 08/31/2013 zurück.
`monthsend(4, '10/19/2013', -1)`	Gibt 08/31/2013 zurück.
`monthsend(4, '10/19/2013', 0, 2)`	Gibt 01/31/2014 zurück. Weil der Start des Jahres auf den Monat 2 verschoben wird.

Beispiel 1 – einfaches Beispiel

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz, der eine Reihe von Transaktionen für 2022 enthält, wird in eine Tabelle namens „`Transactions`“ geladen.
- Datumsfeld, das im Format `(MM/DD/YYYY)` der Systemvariablen `DateFormat` bereitgestellt wird.
- Ein vorangehender load-Befehl, der Folgendes enthält:

- Die Funktion `monthsend`, die als Feld „`bi_monthly_end`“ festgelegt ist. Damit werden Transaktionen in Zweimonatssegmente gruppiert.
- Die Funktion `timestamp`, die den Startzeitstempel des Segments für jede Transaktion zurückgibt.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
    *,
    monthsend(2,date) as bi_monthly_end,
    timestamp(monthsend(2,date)) as bi_monthly_end_timestamp
    ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/22/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

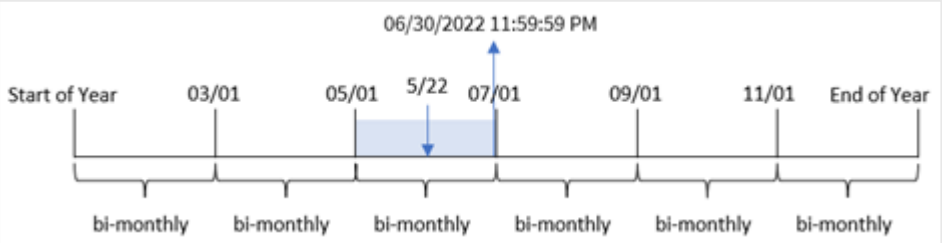
- `id`
- `date`
- `bi_monthly_end`
- `bi_monthly_end_timestamp`

Ergebnistabelle

id	date	bi_monthly_end	bi_monthly_end_timestamp
8188	1/7/2022	02/28/2022	2/28/2022 11:59:59 PM
8189	1/19/2022	02/28/2022	2/28/2022 11:59:59 PM
8190	2/5/2022	02/28/2022	2/28/2022 11:59:59 PM
8191	2/28/2022	02/28/2022	2/28/2022 11:59:59 PM
8192	3/16/2022	04/30/2022	4/30/2022 11:59:59 PM
8193	4/1/2022	04/30/2022	4/30/2022 11:59:59 PM
8194	5/7/2022	06/30/2022	6/30/2022 11:59:59 PM
8195	5/22/2022	06/30/2022	6/30/2022 11:59:59 PM
8196	6/15/2022	06/30/2022	6/30/2022 11:59:59 PM
8197	6/26/2022	06/30/2022	6/30/2022 11:59:59 PM
8198	7/9/2022	08/31/2022	8/31/2022 11:59:59 PM
8199	7/22/2022	08/31/2022	8/31/2022 11:59:59 PM
8200	7/23/2022	08/31/2022	8/31/2022 11:59:59 PM
8201	7/27/2022	08/31/2022	8/31/2022 11:59:59 PM
8202	8/2/2022	08/31/2022	8/31/2022 11:59:59 PM
8203	8/8/2022	08/31/2022	8/31/2022 11:59:59 PM
8204	8/19/2022	08/31/2022	8/31/2022 11:59:59 PM
8205	9/26/2022	10/31/2022	10/31/2022 11:59:59 PM
8206	10/14/2022	10/31/2022	10/31/2022 11:59:59 PM
8207	10/29/2022	10/31/2022	10/31/2022 11:59:59 PM

Das Feld „bi_monthly_end“ wird in der vorangehenden load-Anweisung mithilfe der Funktion `monthsend()` erstellt. Das erste angegebene Argument ist `2`. Damit wird das Jahr in Zweimonatssegmente unterteilt. Das zweite Argument identifiziert, welches Feld ausgewertet wird.

Diagramm der Funktion `monthsend` mit Zweimonatssegmenten.



Transaktion 8195 findet am 22. Mai statt. Die Funktion `monthsend()` unterteilt das Jahr zunächst in Zweimonatssegmente. Transaktion 8195 fällt in das Segment zwischen dem 1. Mai und dem 30. Juni. Als Ergebnis gibt die Funktion die letzte Millisekunde dieses Segments zurück, 06/30/2022 11:59:59 PM.

Beispiel 2 – period_no

Ladeskript und Ergebnisse

Übersicht

Es werden derselbe Datensatz und dasselbe Szenario wie im ersten Beispiel verwendet.

In diesem Beispiel besteht die Aufgabe darin, ein Feld „`prev_bi_monthly_end`“ zu erstellen, das die erste Millisekunde des Zweimonatssegments zurückgibt, bevor die Transaktion stattfand.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
    *,
    monthsend(2,date,-1) as prev_bi_monthly_end,
    timestamp(monthsend(2,date,-1)) as prev_bi_monthly_end_timestamp
    ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/22/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

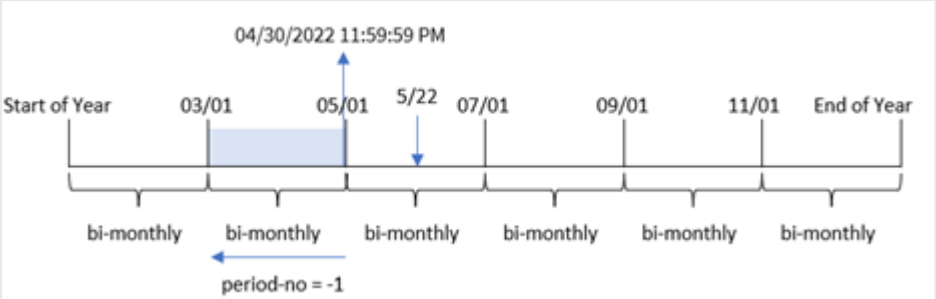
- `id`
- `date`
- `prev_bi_monthly_end`
- `prev_bi_monthly_end_timestamp`

Ergebnistabelle

id	date	prev_bi_monthly_end	prev_bi_monthly_end_timestamp
8188	1/7/2022	12/31/2021	12/31/2021 11:59:59 PM
8189	1/19/2022	12/31/2021	12/31/2021 11:59:59 PM
8190	2/5/2022	12/31/2021	12/31/2021 11:59:59 PM
8191	2/28/2022	12/31/2021	12/31/2021 11:59:59 PM
8192	3/16/2022	02/28/2022	2/28/2022 11:59:59 PM
8193	4/1/2022	02/28/2022	2/28/2022 11:59:59 PM
8194	5/7/2022	04/30/2022	4/30/2022 11:59:59 PM
8195	5/22/2022	04/30/2022	4/30/2022 11:59:59 PM
8196	6/15/2022	04/30/2022	4/30/2022 11:59:59 PM
8197	6/26/2022	04/30/2022	4/30/2022 11:59:59 PM
8198	7/9/2022	06/30/2022	6/30/2022 11:59:59 PM
8199	7/22/2022	06/30/2022	6/30/2022 11:59:59 PM
8200	7/23/2022	06/30/2022	6/30/2022 11:59:59 PM
8201	7/27/2022	06/30/2022	6/30/2022 11:59:59 PM
8202	8/2/2022	06/30/2022	6/30/2022 11:59:59 PM
8203	8/8/2022	06/30/2022	6/30/2022 11:59:59 PM
8204	8/19/2022	06/30/2022	6/30/2022 11:59:59 PM
8205	9/26/2022	08/31/2022	8/31/2022 11:59:59 PM
8206	10/14/2022	08/31/2022	8/31/2022 11:59:59 PM
8207	10/29/2022	08/31/2022	8/31/2022 11:59:59 PM

Da `-1` als Argument `period_no` in der Funktion `monthsend()` verwendet wird, gibt die Funktion, nachdem ein Jahr zunächst in Zweimonatssegmente unterteilt wurde, die letzte Millisekunde des vorherigen Zweimonatssegments vor einer Transaktion zurück.

Diagramm der Funktion `monthsend`, die das vorherige Zweimonatssegment zurückgibt.



Transaktion 8195 fällt in das Segment zwischen dem Mai und Juni. Infolgedessen lag das Zweimonatssegment zwischen dem 1. März und dem 30. April, und die Funktion gibt die letzte Millisekunde dieses Segments zurück, 04/30/2022 11:59:59 PM.

Beispiel 3 – first_month_of_year

Ladeskript und Ergebnisse

Übersicht

Es werden derselbe Datensatz und dasselbe Szenario wie im ersten Beispiel verwendet.

In diesem Beispiel legt die Organisationsrichtlinie fest, dass April der erste Monat des Geschäftsjahres ist.

Es wird ein Feld „`bi_monthly_end`“, das Transaktionen in Zweimonatssegmenten gruppiert und den Zeitstempel für die letzte Millisekunde des Segments für jede Transaktion zurückgibt.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
    *,
    monthsend(2,date,0,4) as bi_monthly_end,
    timestamp(monthsend(2,date,0,4)) as bi_monthly_end_timestamp
    ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/22/2022,87.21
8196,6/15/2022,95.93
```

```
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `id`
- `date`
- `bi_monthly_end`
- `bi_monthly_end_timestamp`

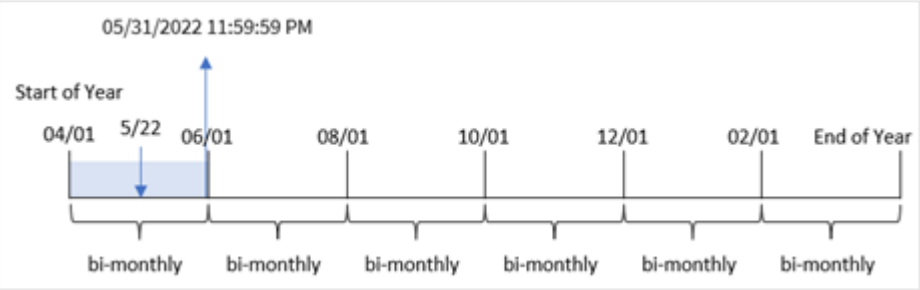
Ergebnistabelle

id	date	bi_monthly_end	bi_monthly_end_timestamp
8188	1/7/2022	01/31/2022	1/31/2022 11:59:59 PM
8189	1/19/2022	01/31/2022	1/31/2022 11:59:59 PM
8190	2/5/2022	03/31/2022	3/31/2022 11:59:59 PM
8191	2/28/2022	03/31/2022	3/31/2022 11:59:59 PM
8192	3/16/2022	03/31/2022	3/31/2022 11:59:59 PM
8193	4/1/2022	05/31/2022	5/31/2022 11:59:59 PM
8194	5/7/2022	05/31/2022	5/31/2022 11:59:59 PM
8195	5/22/2022	05/31/2022	5/31/2022 11:59:59 PM
8196	6/15/2022	07/31/2022	7/31/2022 11:59:59 PM
8197	6/26/2022	07/31/2022	7/31/2022 11:59:59 PM
8198	7/9/2022	07/31/2022	7/31/2022 11:59:59 PM
8199	7/22/2022	07/31/2022	7/31/2022 11:59:59 PM
8200	7/23/2022	07/31/2022	7/31/2022 11:59:59 PM
8201	7/27/2022	07/31/2022	7/31/2022 11:59:59 PM
8202	8/2/2022	09/30/2022	9/30/2022 11:59:59 PM

id	date	bi_monthly_end	bi_monthly_end_timestamp
8203	8/8/2022	09/30/2022	9/30/2022 11:59:59 PM
8204	8/19/2022	09/30/2022	9/30/2022 11:59:59 PM
8205	9/26/2022	09/30/2022	9/30/2022 11:59:59 PM
8206	10/14/2022	11/30/2022	11/30/2022 11:59:59 PM
8207	10/29/2022	11/30/2022	11/30/2022 11:59:59 PM

Da 4 als Argument `first_month_of_year` in der Funktion `monthsend()` verwendet wird, beginnt die Funktion das Jahr am 1. April und unterteilt dann das Jahr in Zweimonatssegmente: Apr-Mai, Jun-Jul, Aug-Sep, Okt-Nov, Dez-Jan, Feb-Mär.

Diagramm der Funktion `monthsend` mit April als erstem Monat des Jahres.



Transaktion 8195 fand am 22. Mai statt und fällt in das Segment zwischen dem 1. April und dem 31. Mai. Als Ergebnis gibt die Funktion die letzte Millisekunde dieses Segments zurück, 05/31/2022 11:59:59 PM.

Beispiel 4 – Diagrammobjektbeispiel

Ladeskript und Diagrammformel

Übersicht

Es werden derselbe Datensatz und dasselbe Szenario wie im ersten Beispiel verwendet. In diesem Beispiel wird jedoch der unveränderte Datensatz in die App geladen.

In diesem Beispiel besteht die Aufgabe darin, eine Berechnung zu erstellen, die Transaktionen in Zweimonatssegmenten gruppiert und den Zeitstempel für die letzte Millisekunde des Segments für jede Transaktion als Kennzahl in einem Diagrammobjekt für eine App zurückgibt.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
```

```
8188,2/19/2022,37.23
8189,3/7/2022,17.17
8190,3/30/2022,88.27
8191,4/5/2022,57.42
8192,4/16/2022,53.80
8193,5/1/2022,82.06
8194,5/7/2022,40.39
8195,5/22/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu:

`date`

Um den Zeitstempel für die letzte Millisekunde des Zweimonatssegments abzurufen, in dem die Transaktion stattfand, erstellen Sie die folgenden Kennzahlen

- `=monthsEnd(2,date)`
- `=timestamp(monthsend(2,date))`

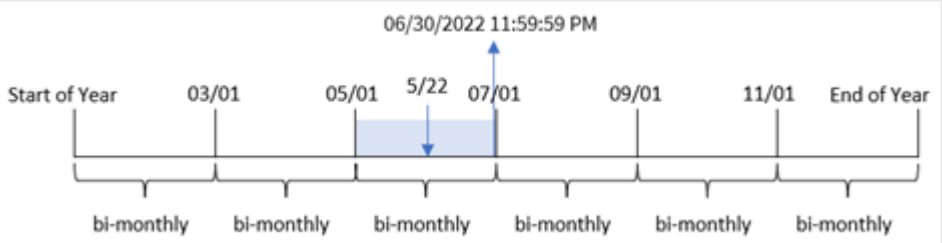
Ergebnistabelle

id	date	=monthsend(2,date)	=timestamp(monthsend(2,date))
8188	1/7/2022	02/28/2022	2/28/2022 11:59:59 PM
8189	1/19/2022	02/28/2022	2/28/2022 11:59:59 PM
8190	2/5/2022	02/28/2022	2/28/2022 11:59:59 PM
8191	2/28/2022	02/28/2022	2/28/2022 11:59:59 PM
8192	3/16/2022	04/30/2022	4/30/2022 11:59:59 PM
8193	4/1/2022	04/30/2022	4/30/2022 11:59:59 PM
8194	5/7/2022	06/30/2022	6/30/2022 11:59:59 PM
8195	5/22/2022	06/30/2022	6/30/2022 11:59:59 PM

id	date	=monthsend(2,date)	=timestamp(monthsend(2,date))
8196	6/15/2022	06/30/2022	6/30/2022 11:59:59 PM
8197	6/26/2022	06/30/2022	6/30/2022 11:59:59 PM
8198	7/9/2022	08/31/2022	8/31/2022 11:59:59 PM
8199	7/22/2022	08/31/2022	8/31/2022 11:59:59 PM
8200	7/23/2022	08/31/2022	8/31/2022 11:59:59 PM
8201	7/27/2022	08/31/2022	8/31/2022 11:59:59 PM
8202	8/2/2022	08/31/2022	8/31/2022 11:59:59 PM
8203	8/8/2022	08/31/2022	8/31/2022 11:59:59 PM
8204	8/19/2022	08/31/2022	8/31/2022 11:59:59 PM
8205	9/26/2022	10/31/2022	10/31/2022 11:59:59 PM
8206	10/14/2022	10/31/2022	10/31/2022 11:59:59 PM
8207	10/29/2022	10/31/2022	10/31/2022 11:59:59 PM

Das Feld „`bi_monthly_end`“ wird als Kennzahl im Diagrammobjekt mithilfe der Funktion `monthsend()` erstellt. Das erste angegebene Argument ist `2`. Damit wird das Jahr in Zweimonatssegmente unterteilt. Das zweite Argument identifiziert, welches Feld ausgewertet wird.

Diagramm der Funktion `monthsend` mit Zweimonatssegmenten.



Transaktion 8195 findet am 22. Mai statt. Die Funktion `monthsend()` unterteilt das Jahr zunächst in Zweimonatssegmente. Transaktion 8195 fällt in das Segment zwischen dem 1. Mai und dem 30. Juni. Als Ergebnis gibt die Funktion die erste Millisekunde dieses Segments zurück, 06/30/2022 11:59:59 PM.

Beispiel 5 – Szenario

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

In diesem Beispiel wird ein Datensatz in eine Tabelle namens „`Employee_Expenses`“ geladen. Die Tabelle enthält die folgenden Felder:

- Mitarbeiter-IDs
- Mitarbeiternamen
- Die durchschnittlichen täglichen Spesenanträge pro Mitarbeiter.

Der Endbenutzer möchte ein Diagramm, das nach Mitarbeiter-ID und Mitarbeiternamen die geschätzten Spesenanträge anzeigt, die für den Rest eines von ihm ausgewählten Zeitraums noch anfallen. Das Geschäftsjahr beginnt im Januar.

Ladeskript

```
SET vPeriod = 1;

Employee_Expenses:
Load
*
Inline
[
employee_id,employee_name,avg_daily_claim
182,Mark, $15
183,Deryck, $12.5
184,Dexter, $12.5
185,Sydney,$27
186,Agatha,$18
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein neues Arbeitsblatt.

Am Beginn des Ladeskripts wurde eine Variable (`vPeriod`) erstellt, die an die Variableneingabesteuerung gebunden wird.

Gehen Sie folgendermaßen vor:

1. Klicken Sie im Extras-Fenster auf **Benutzerdefinierte Objekte**.
2. Wählen Sie **Qlik Dashboard Bundle** aus und erstellen Sie ein Objekt **Variableneingabe**.
3. Geben Sie einen Titel für das Diagrammobjekt ein.
4. Wählen Sie unter **Variable** den Eintrag **vPeriod** als den Namen aus und legen Sie das Objekt so fest, dass es als **Dropdown** angezeigt wird.
5. Klicken Sie unter **Werte** auf **Dynamisch**. Geben Sie Folgendes ein:
   `='1~month|2~bi-month|3~quarter|4~tertial|6~half-year'.`

Erstellen Sie eine neue Tabelle und die folgenden Felder als Dimensionen:

- `employee_id`
- `employee_name`

Erstellen Sie die folgende Kennzahl, um die kumulierten Zinsen zu berechnen:

```
=floor(monthsend($(vPeriod),today(1))-today(1))*avg_daily_claim
```

Diese Kennzahl ist dynamisch und ergibt unterschiedliche Tabellenergebnisse, je nach dem Datum, an dem Sie die Daten laden.

Legen Sie das **Zahlenformat** der Kennzahl auf **Währung** fest.

Ergebnistabelle

employee_id	employee_name	=floor(monthsend($(vPeriod),today(1))-today(1))*avg_daily_claim
182	Mark	$1410.00
183	Deryck	$1175.00
184	Dexter	$1175.00
185	Sydney	$2538.00
186	Agatha	$1692.00

Die Funktion `monthsend()` verwendet die Benutzereingabe als erstes Argument und das aktuelle Datum als zweites Argument. Damit wird das Enddatum für den vom Benutzer ausgewählten Zeitraum zurückgegeben. Dann gibt die Formel die Anzahl der im ausgewählten Zeitraum verbleibenden Tage zurück, indem das aktuelle Datum vom betreffenden Enddatum abgezogen wird.

Dieser Wert wird dann mit den durchschnittlichen täglichen Spesenanträgen der einzelnen Mitarbeitern multipliziert, um den geschätzten Spesenbetrag pro Mitarbeiter für die verbleibenden Tage des Zeitraums zu berechnen.

monthsname

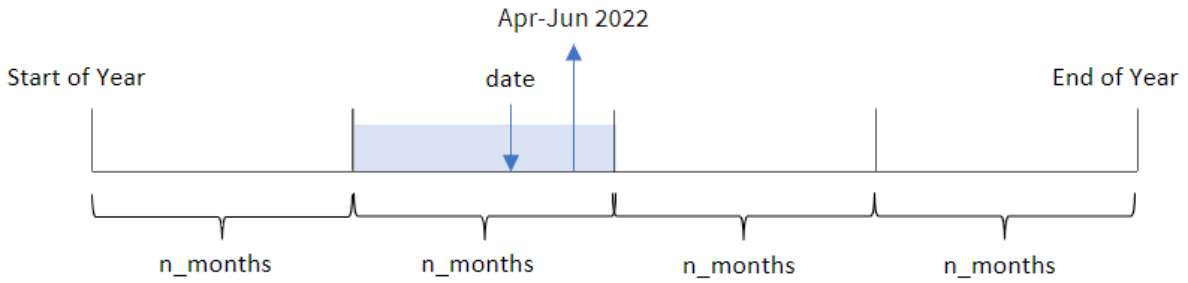
Diese Funktion liefert einen Anzeigewert, der den Bereich der Monate des Zeitraums (formatiert nach der **MonthNames**-Skriptvariable) sowie das Jahr darstellt. Der zugrunde liegende numerische Wert entspricht dem Zeitstempel der ersten Millisekunde des Monats, Zweimonatszeitraums, Quartals, Viermonatszeitraums oder Halbjahrs, in dem ein Basisdatum liegt.

Syntax:

```
MonthsName(n_months, date[, period_no[, first_month_of_year]])
```

Rückgabe Datentyp: dual

Diagramm der Funktion „monthsname“



Die Funktion `monthsname()` unterteilt das Jahr in Segmente, gestützt auf das angegebene Argument `n_months`. Dann wertet sie das Segment aus, zu dem jedes angegebene `date` gehört, und gibt die Namen des Start- und Endmonats für dieses Segment sowie das Jahr zurück. Die Funktion bietet auch die Möglichkeit, diese Grenzen aus vorangehenden oder darauffolgenden Segmenten zurückzugeben und neu zu definieren, welcher der erste Monat des Jahres ist.

Die folgenden Segmente des Jahres sind in der Funktion als `n_month`-Argumente verfügbar:

Mögliche `n_month`-Argumente

Zeiträume	**Anzahl der Monate**
Monat	1
Zweimonatszeitraum	2
Quartal	3
Viermonatszeitraum	4
Halbjahr	6

Argumente

Argument	**Beschreibung**
n_months	Die Anzahl der Monate, die den Zeitraum definiert. Eine Ganzzahl oder eine Formel, die eine Ganzzahl mit einem der folgenden Werte ergibt: 1 (entspricht der Funktion inmonth()), 2 (Zweimonatszeitraum), 3 (entspricht der Funktion inquarter()), 4 (Viermonatszeitraum) oder 6 (Halbjahr).
date	Datum oder Zeitstempel für die Evaluierung.
period_no	Mit **period_no**, einer ganze Zahl oder einer Formel, die eine ganze Zahl ergibt, kann ein anderer Beginn für den Zeitraum festgelegt werden, wobei 0 für den Zeitraum steht, der **base_date** enthält. Negative Werte von **period_no** stehen für vorangehende, positive Werte für nachfolgende Zeiträume.

Argument	Beschreibung
first_ month_of_ year	Wenn Sie mit abweichenden Geschäftsjahren arbeiten möchten, definieren Sie mit einer Zahl zwischen 2 und 12 für **first_month_of_year** einen anderen Startmonat.

Verwendung

Die Funktion `monthsname()` ist nützlich, wenn Sie dem Benutzer die Möglichkeit geben möchten, Aggregierungen für einen selbst gewählten Zeitraum zu vergleichen. Sie können beispielsweise eine Eingabevariable bereitstellen, damit der Benutzer den Gesamtumsatz von Produkten nach Monat, Quartal oder Halbjahr anzeigen kann.

Diese Dimensionen können entweder im Ladeskript erstellt werden, indem die Funktion als Feld in einer Master-Kalender-Tabelle hinzugefügt wird, oder indem die Dimension direkt in einem Diagramm als berechnete Dimension erstellt wird.

Funktionsbeispiele

Beispiel	Ergebnis
`monthsname(4, '10/19/2013')`	Gibt „Sep-Dec 2013“ zurück. In diesem und den anderen Beispielen ist die Anweisung **SET Monthnames** auf Jan;Feb;Mar usw. festgelegt.
`monthsname(4, '10/19/2013', -1)`	Gibt „May-Aug 2013“ zurück.
`monthsname(4, '10/19/2013', 0, 2)`	Gibt „Oct-Jan 2014“ zurück, da festgelegt ist, dass das Jahr im Monat 2 beginnt. Daher endet der Viermonatszeitraum im ersten Monat des Folgejahres.

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Beispiel 1 – einfaches Beispiel

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz enthält eine Reihe von Transaktionen für 2022 und wird in eine Tabelle namens `Transactions` geladen.
- Das Datumsfeld wird im Format der Systemvariablen `DateFormat` (MM/TT/JJJJ) bereitgestellt.
- Die Erstellung eines Felds `bi_monthly_range`, das Transaktionen in Zweimonatssegmenten gruppiert und die Grenznamen dieses Segments für jede Transaktion zurückgibt.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        monthsname(2,date) as bi_monthly_range
        ;
Load
*
Inline
[
id,date,amount
8188,2/19/2022,37.23
8189,3/7/2022,17.17
8190,3/30/2022,88.27
8191,4/5/2022,57.42
8192,4/16/2022,53.80
8193,5/1/2022,82.06
8194,5/7/2022,40.39
8195,5/22/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

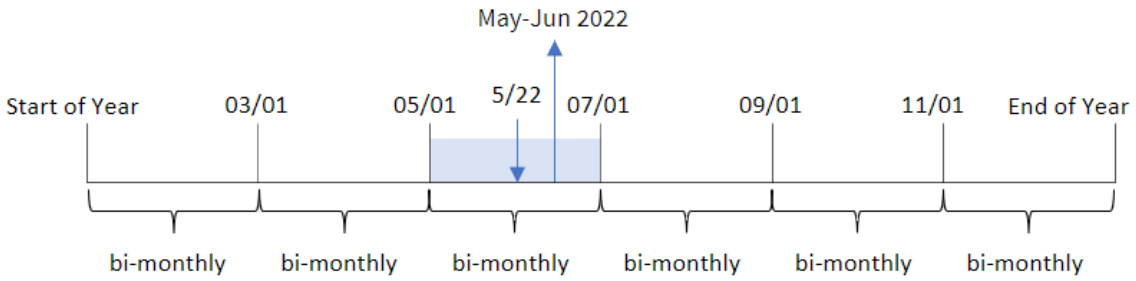
- `date`
- `bi_monthly_range`

Ergebnistabelle

date	bi_monthly_range
2/19/2022	Jan-Feb 2022
3/7/2022	Mar-Apr 2022
3/30/2022	Mar-Apr 2022
4/5/2022	Mar-Apr 2022
4/16/2022	Mar-Apr 2022
5/1/2022	May-Jun 2022
5/7/2022	May-Jun 2022
5/22/2022	May-Jun 2022
6/15/2022	May-Jun 2022
6/26/2022	May-Jun 2022
7/9/2022	Jul-Aug 2022
7/22/2022	Jul-Aug 2022
7/23/2022	Jul-Aug 2022
7/27/2022	Jul-Aug 2022
8/2/2022	Jul-Aug 2022
8/8/2022	Jul-Aug 2022
8/19/2022	Jul-Aug 2022
9/26/2022	Sep-Oct 2022
10/14/2022	Sep-Oct 2022
10/29/2022	Sep-Oct 2022

Das Feld „`bi_monthly_range`“ wird im vorangehenden load-Befehl mithilfe der Funktion `monthsname()` erstellt. Das erste angegebene Argument ist 2. Damit wird das Jahr in Zweimonatssegmente unterteilt. Das zweite Argument identifiziert, welches Feld ausgewertet wird.

Diagramm der Funktion „monthsname“, einfaches Beispiel



Transaktion 8195 findet am 22. Mai statt. Die Funktion `monthsname()` unterteilt das Jahr zunächst in Zweimonatssegmente. Transaktion 8195 fällt in das Segment zwischen dem 1. Mai und dem 30. Juni. Daher gibt die Funktion diese Monate im Systemvariablenformat `MonthNames` sowie das Jahr zurück: May-Jun 2022.

Beispiel 2 – period_no

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Derselbe Inline-Datensatz und dasselbe Szenario wie im ersten Beispiel.
- Die Erstellung eines Felds `prev_bi_monthly_range`, das Transaktionen in Zweimonatssegmenten gruppiert und die Grenznamen des vorherigen Segments für jede Transaktion zurückgibt.

Fügen Sie bei Bedarf hier weiteren Text mit Listen usw. hinzu.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        MonthsName(2,date,-1) as prev_bi_monthly_range
        ;
Load
*
Inline
[
id,date,amount
8188,2/19/2022,37.23
8189,3/7/2022,17.17
8190,3/30/2022,88.27
8191,4/5/2022,57.42
8192,4/16/2022,53.80
```

```
8193,5/1/2022,82.06
8194,5/7/2022,40.39
8195,5/22/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `prev_bi_monthly_range`

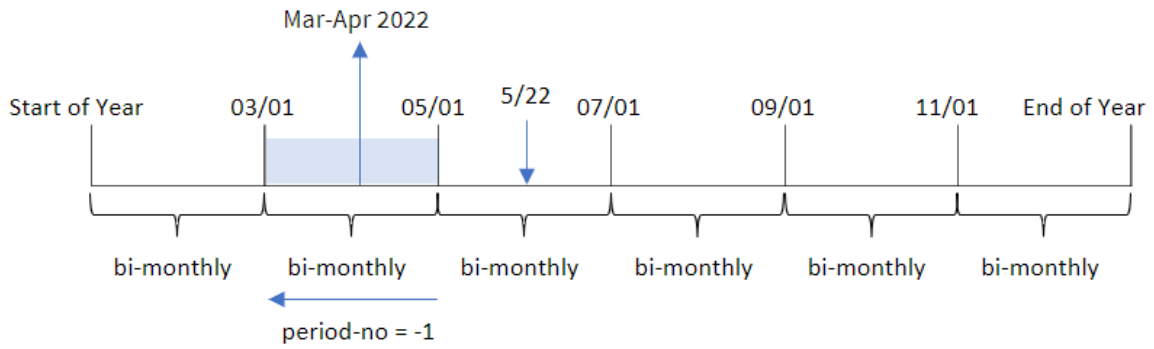
Ergebnistabelle

date	**prev_bi_monthly_range**
2/19/2022	Nov-Dec 2021
3/7/2022	Jan-Feb 2022
3/30/2022	Jan-Feb 2022
4/5/2022	Jan-Feb 2022
4/16/2022	Jan-Feb 2022
5/1/2022	Mar-Apr 2022
5/7/2022	Mar-Apr 2022
5/22/2022	Mar-Apr 2022
6/15/2022	Mar-Apr 2022
6/26/2022	Mar-Apr 2022
7/9/2022	May-Jun 2022
7/22/2022	May-Jun 2022
7/23/2022	May-Jun 2022
7/27/2022	May-Jun 2022

date	prev_bi_monthly_range
8/2/2022	May-Jun 2022
8/8/2022	May-Jun 2022
8/19/2022	May-Jun 2022
9/26/2022	Jul-Aug 2022
10/14/2022	Jul-Aug 2022
10/29/2022	Jul-Aug 2022

In diesem Beispiel wird -1 als Argument `period_no` in der Funktion `monthsname()` verwendet. Nachdem ein Jahr zunächst in Zweimonatssegmente unterteilt wurde, gibt die Funktion dann die Segmentgrenzen vor dem Zeitpunkt zurück, an dem eine Transaktion stattfindet.

Diagramm der Funktion „monthsname“, Beispiel „period_no“



Transaktion 8195 fällt in das Segment zwischen dem Mai und Juni. Daher lag das vorherige Zweimonatssegment zwischen dem 1. März und dem 30. April, und die Funktion gibt „Mar-Apr 2022“ zurück.

Beispiel 3 – first_month_of_year

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Derselbe Inline-Datensatz und dasselbe Szenario wie im ersten Beispiel.
- Die Erstellung eines weiteren Felds `bi_monthly_range`, das Transaktionen in Zweimonatssegmenten gruppiert und die Segmentgrenzen für jede Transaktion zurückgibt.

In diesem Beispiel müssen wir aber zudem April als ersten Monat im Geschäftsjahr festlegen.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        MonthsName(2,date,0,4) as bi_monthly_range
        ;
Load
*
Inline
[
id,date,amount
8188,2/19/2022,37.23
8189,3/7/2022,17.17
8190,3/30/2022,88.27
8191,4/5/2022,57.42
8192,4/16/2022,53.80
8193,5/1/2022,82.06
8194,5/7/2022,40.39
8195,5/22/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `bi_monthly_range`

Ergebnistabelle

date	bi_monthly_range
2/19/2022	Feb-Mar 2021
3/7/2022	Feb-Mar 2021
3/30/2022	Feb-Mar 2021

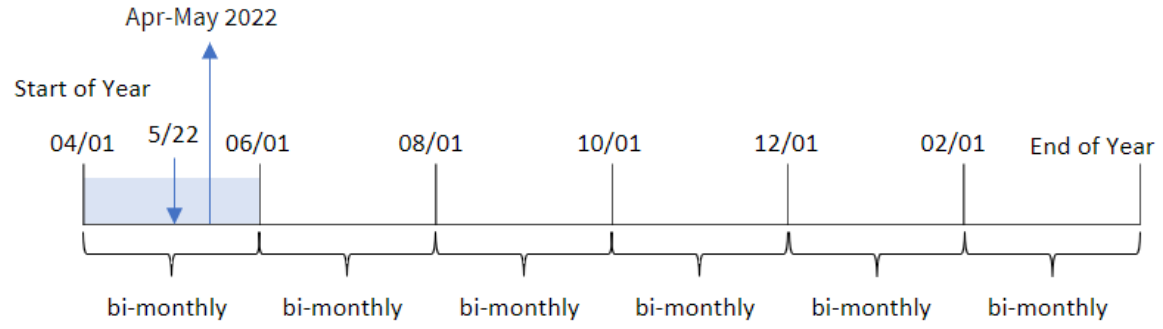
date	bi_monthly_range
4/5/2022	Apr-May 2022
4/16/2022	Apr-May 2022
5/1/2022	Apr-May 2022
5/7/2022	Apr-May 2022
5/22/2022	Apr-May 2022
6/15/2022	Jun-Jul 2022
6/26/2022	Jun-Jul 2022
7/9/2022	Jun-Jul 2022
7/22/2022	Jun-Jul 2022
7/23/2022	Jun-Jul 2022
7/27/2022	Jun-Jul 2022
8/2/2022	Aug-Sep 2022
8/8/2022	Aug-Sep 2022
8/19/2022	Aug-Sep 2022
9/26/2022	Aug-Sep 2022
10/14/2022	Oct-Nov 2022
10/29/2022	Oct-Nov 2022

Indem 4 als Argument `first_month_of_year` in der Funktion `monthsname()` verwendet wird, beginnt die Funktion das Jahr am 1. April und unterteilt dann das Jahr in Zweimonatssegmente: Apr-May,Jun-Jul,Aug-Sep,Oct-Nov,Dec-Jan,Feb-Mar.

Absatztext für Ergebnisse.

Transaktion 8195 fand am 22. Mai statt und fällt in das Segment zwischen dem 1. April und dem 31. Mai. Daher gibt die Funktion „Apr-May 2022“ zurück..

Diagramm der Funktion „monthsname“, Beispiel „first_month_of_year“



Beispiel 4 – Diagrammobjektbeispiel

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript verwendet den gleichen Inline-Datensatz und das gleiche Szenario wie das erste Beispiel. In diesem Beispiel wird jedoch der unveränderte Datensatz in die Anwendung geladen. Die Berechnung, die Transaktionen in Zweimonatssegmenten gruppiert und die Segmentgrenzen für jede Transaktion zurückgibt, wird als Kennzahl in einem Diagrammobjekt der Anwendung erstellt.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8188,2/19/2022,37.23
8189,3/7/2022,17.17
8190,3/30/2022,88.27
8191,4/5/2022,57.42
8192,4/16/2022,53.80
8193,5/1/2022,82.06
8194,5/7/2022,40.39
8195,5/22/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu:`date`.

Erstellen Sie die folgende Kennzahl:

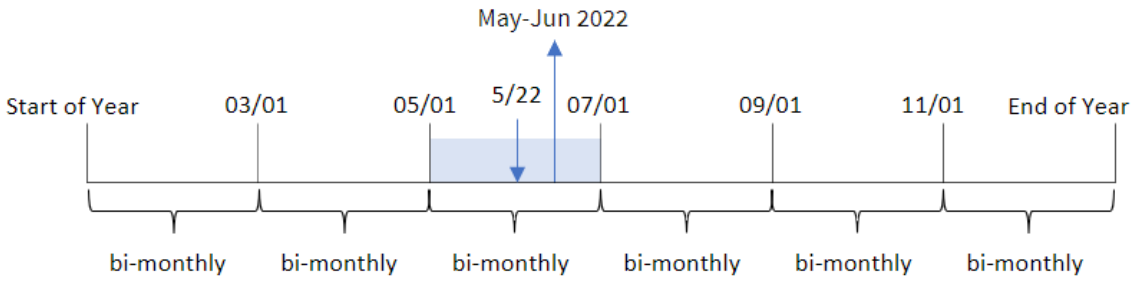
```
=monthsname(2,date)
```

Ergebnistabelle

date	=monthsname(2,date)
2/19/2022	Jan-Feb 2022
3/7/2022	Mar-Apr 2022
3/30/2022	Mar-Apr 2022
4/5/2022	Mar-Apr 2022
4/16/2022	Mar-Apr 2022
5/1/2022	May-Jun 2022
5/7/2022	May-Jun 2022
5/22/2022	May-Jun 2022
6/15/2022	May-Jun 2022
6/26/2022	May-Jun 2022
7/9/2022	Jul-Aug 2022
7/22/2022	Jul-Aug 2022
7/23/2022	Jul-Aug 2022
7/27/2022	Jul-Aug 2022
8/2/2022	Jul-Aug 2022
8/8/2022	Jul-Aug 2022
8/19/2022	Jul-Aug 2022
9/26/2022	Sep-Oct 2022
10/14/2022	Sep-Oct 2022
10/29/2022	Sep-Oct 2022

Das Feld `bi_monthly_range` wird als Kennzahl im Diagrammobjekt mithilfe der Funktion `monthsname()` erstellt. Das erste angegebene Argument ist 2. Damit wird das Jahr in Zweimonatssegmente unterteilt. Das zweite Argument identifiziert, welches Feld ausgewertet wird.

Diagramm der Funktion „monthsname“, Diagrammobjektbeispiel



Transaktion 8195 findet am 22. Mai statt. Die Funktion `monthsname()` unterteilt das Jahr zunächst in Zweimonatssegmente. Transaktion 8195 fällt in das Segment zwischen dem 1. Mai und dem 30. Juni. Daher gibt die Funktion diese Monate im Systemvariablenformat `MonthNames` sowie das Jahr zurück: May-Jun 2022.

Beispiel 5 – Szenario

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz, der Transaktionen für 2022 enthält und der in eine Tabelle namens `Transactions` geladen wird.
- Das Datumsfeld wurde im Format der Systemvariablen `DateFormat` (MM/TT/JJJJ) bereitgestellt.

Der Endbenutzer möchte ein Diagrammobjekt, das den Gesamtumsatz nach einem selbst gewählten Zeitraum anzeigt. Dies ist selbst dann möglich, wenn diese Dimension nicht im Datenmodell verfügbar ist. In diesem Fall wird die Funktion `monthsname()` als berechnete Dimension verwendet, die durch eine Variableneingabesteuerung dynamisch geändert wird.

Ladeskript

```
SET vPeriod = 1;
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8188,'1/7/2022',17.17
8189,'1/19/2022',37.23
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
```

```
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt.

Am Beginn des Ladeskripts wurde eine Variable (`vPeriod`) erstellt, die an die Variableneingabesteuerung gebunden wird. Dann wird die Variable als benutzerdefiniertes Objekt auf dem Arbeitsblatt konfiguriert.

Gehen Sie folgendermaßen vor:

1. Klicken Sie im Extras-Fenster auf **Benutzerdefinierte Objekte**.
2. Wählen Sie **Qlik Dashboard Bundle** aus und erstellen Sie ein Objekt **Variableneingabe**.
3. Geben Sie einen Titel für das Diagrammobjekt ein.
4. Wählen Sie unter **Variable** den Eintrag **vPeriod** als den Namen aus und legen Sie das Objekt so fest, dass es als **Dropdown** angezeigt wird.
5. Konfigurieren Sie unter **Werte** das Objekt für die Verwendung von dynamischen Werten. Geben Sie Folgendes ein:
   `='1~month|2~bi-month|3~quarter|4~tertial|6~half-year'`

Erstellen Sie dann die Ergebnistabelle.

Gehen Sie folgendermaßen vor:

1. Erstellen Sie eine neue Tabelle und fügen Sie die folgende berechnete Dimension hinzu:
   `=monthsname($(vPeriod),date)`
2. Fügen Sie diese Kennzahl hinzu, um den Gesamtumsatz zu berechnen:
   `=sum(amount)`
3. Legen Sie das **Zahlenformat** der Kennzahl auf **Währung** fest. Klicken Sie auf ✔ **Bearbeitung fertig**. Jetzt können Sie das in der Tabelle angezeigte Datum ändern, indem Sie das Zeitsegment im Variablenobjekt anpassen.

Die Ergebnistabelle sieht wie folgt aus, wenn die Option `tertial` ausgewählt wird:

Ergebnistabelle

monthsname($(vPeriod),date)	=sum(amount)
Jan-Apr 2022	253.89
May-Aug 2022	713.58
Sep-Dec 2022	248.12

monthsstart

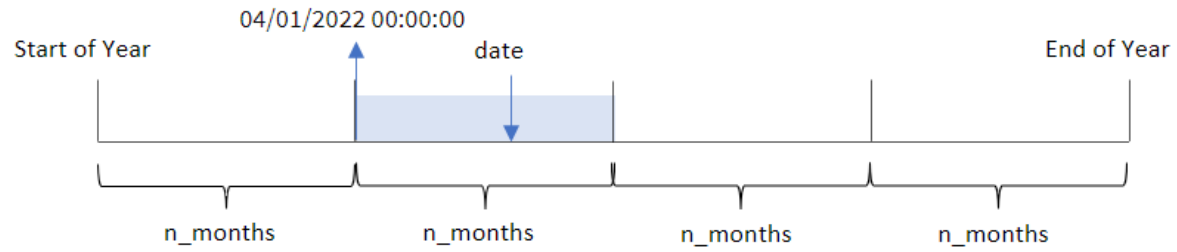
Diese Funktion gibt einen Wert zurück, der einem Zeitstempel für die erste Millisekunde im Monat, Zweimonatszeitraum, Quartal, Viermonatszeitraum oder Halbjahr entspricht, in dem ein Basisdatum liegt. Es lässt sich auch der Zeitstempel für einen vorhergehenden oder nachfolgenden Zeitraum bestimmen.Das Standardausgabeformat ist das im Skript definierte **DateFormat**.

Syntax:

```
MonthsStart(n_months, date[, period_no [, first_month_of_year]])
```

Rückgabe Datentyp: dual

Diagramm der Funktion `monthsstart()`



Die Funktion `monthsstart()` unterteilt das Jahr in Segmente, gestützt auf das angegebene Argument `n_months`. Dann wertet sie aus, in welches Segment jedes angegebene Datum fällt, und gibt die erste Millisekunde dieses Segments im Datumsformat zurück. Die Funktion bietet auch die Möglichkeit, den Startzeitstempel der vorangehenden oder darauffolgenden Segmente zurückzugeben und neu zu definieren, welcher der erste Monat des Jahres ist.

Die folgenden Segmente des Jahres sind in der Funktion als `n_month`-Argumente verfügbar:

Mögliche `n_month`-Argumente

Zeiträume	Anzahl der Monate
Monat	1
Zweimonatszeitraum	2
Quartal	3

Zeiträume	Anzahl der Monate
Viermonatszeitraum	4
Halbjahr	6

Argumente

Argument	Beschreibung
n_months	Die Anzahl der Monate, die den Zeitraum definiert. Eine Ganzzahl oder eine Formel, die eine Ganzzahl mit einem der folgenden Werte ergibt: 1 (entspricht der Funktion inmonth()), 2 (Zweimonatszeitraum), 3 (entspricht der Funktion inquarter()), 4 (Viermonatszeitraum) oder 6 (Halbjahr).
date	Datum oder Zeitstempel für die Evaluierung.
period_no	Mit **period_no**, einer ganze Zahl oder einer Formel, die eine ganze Zahl ergibt, kann ein anderer Beginn für den Zeitraum festgelegt werden, wobei 0 für den Zeitraum steht, der **base_date** enthält. Negative Werte von **period_no** stehen für vorangehende, positive Werte für nachfolgende Zeiträume.
first_month_of_year	Wenn Sie mit abweichenden Geschäftsjahren arbeiten möchten, definieren Sie mit einer Zahl zwischen 2 und 12 für **first_month_of_year** einen anderen Startmonat.

Verwendung

Die Funktion `monthsstart()` wird in der Regel als Teil einer Formel verwendet, wenn in der Berechnung der Teil eines Zeitraums verwendet werden soll, der noch nicht eingetreten ist. Beispielsweise kann damit eine Eingabevariable angegeben werden, mit der der Benutzer die gesamten während des Monats, Quartals oder Halbjahres aufgelaufenen Zinsen berechnen kann.

Funktionsbeispiele

Beispiel	Ergebnis
`monthsstart(4, '10/19/2013')`	Gibt 09/01/2013 zurück.
`monthsstart(4, '10/19/2013, -1)`	Gibt 05/01/2013 zurück.
`monthsstart(4, '10/19/2013', 0, 2)`	Gibt 10/01/2013 zurück, weil Monat 2 zum Start des Jahres wird.

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Beispiel 1 – keine zusätzlichen Argumente

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz enthält eine Reihe von Transaktionen für 2022 und wird in eine Tabelle namens `Transactions` geladen.
- Das Datumsfeld wird im Format der Systemvariablen `DateFormat` (MM/TT/JJJJ) bereitgestellt.
- Es wird ein Feld `bi_monthly_start` erstellt, das Transaktionen in Zweimonatssegmenten gruppiert und den Startzeitstempel des Segments für jede Transaktion zurückgibt.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        monthsstart(2,date) as bi_monthly_start,
        timestamp(monthsstart(2,date)) as bi_monthly_start_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,2/19/2022,37.23
8189,3/7/2022,17.17
8190,3/30/2022,88.27
8191,4/5/2022,57.42
8192,4/16/2022,53.80
8193,5/1/2022,82.06
8194,5/7/2022,40.39
8195,5/22/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
```

```
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `bi_monthly_start`
- `bi_monthly_start_timestamp`

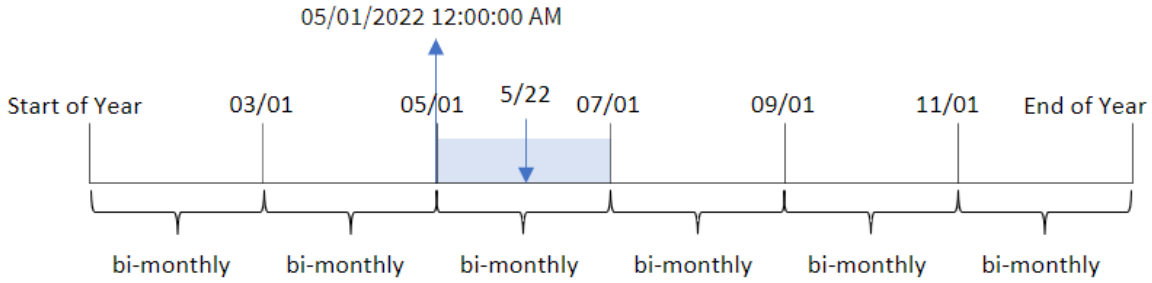
Ergebnistabelle

date	bi_monthly_start	bi_monthly_start_timestamp
2/19/2022	01/01/2022	1/1/2022 12:00:00 AM
3/7/2022	03/01/2022	3/1/2022 12:00:00 AM
3/30/2022	03/01/2022	3/1/2022 12:00:00 AM
4/5/2022	03/01/2022	3/1/2022 12:00:00 AM
4/16/2022	03/01/2022	3/1/2022 12:00:00 AM
5/1/2022	05/01/2022	5/1/2022 12:00:00 AM
5/7/2022	05/01/2022	5/1/2022 12:00:00 AM
5/22/2022	05/01/2022	5/1/2022 12:00:00 AM
6/15/2022	05/01/2022	5/1/2022 12:00:00 AM
6/26/2022	05/01/2022	5/1/2022 12:00:00 AM
7/9/2022	07/01/2022	7/1/2022 12:00:00 AM
7/22/2022	07/01/2022	7/1/2022 12:00:00 AM
7/23/2022	07/01/2022	7/1/2022 12:00:00 AM
7/27/2022	07/01/2022	7/1/2022 12:00:00 AM
8/2/2022	07/01/2022	7/1/2022 12:00:00 AM
8/8/2022	07/01/2022	7/1/2022 12:00:00 AM
8/19/2022	07/01/2022	7/1/2022 12:00:00 AM
9/26/2022	09/01/2022	9/1/2022 12:00:00 AM

date	bi_monthly_start	bi_monthly_start_timestamp
10/14/2022	09/01/2022	9/1/2022 12:00:00 AM
10/29/2022	09/01/2022	9/1/2022 12:00:00 AM

Das Feld „bi_monthly_start“ wird im vorangehenden load-Befehl mithilfe der Funktion `monthsstart()` erstellt. Das erste angegebene Argument ist 2. Damit wird das Jahr in Zweimonatssegmente unterteilt. Das zweite Argument identifiziert, welches Feld ausgewertet wird.

Diagramm der Funktion `monthsstart()`, Beispiel ohne zusätzliche Argumente



Transaktion 8195 findet am 22. Mai statt. Die Funktion `monthsstart()` unterteilt das Jahr zunächst in Zweimonatssegmente. Transaktion 8195 fällt in das Segment zwischen dem 1. Mai und dem 30. Juni. Daher gibt die Funktion die erste Millisekunde dieses Segments zurück, den 1. Mai 2022 um 12:00:00 AM.

Beispiel 2 – period_no

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Derselbe Datensatz und dasselbe Szenario wie im ersten Beispiel.
- Es wird ein Feld `prev_bi_monthly_start` erstellt, das die erste Millisekunde des Zweimonatssegments zurückgibt, bevor die Transaktion stattfand.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        monthsstart(2,date,-1) as prev_bi_monthly_start,
           timestamp(monthsstart(2,date,-1)) as prev_bi_monthly_start_timestamp
           ;
```

```
Load
*
Inline
[
id,date,amount
8188,2/19/2022,37.23
8189,3/7/2022,17.17
8190,3/30/2022,88.27
8191,4/5/2022,57.42
8192,4/16/2022,53.80
8193,5/1/2022,82.06
8194,5/7/2022,40.39
8195,5/22/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `prev_bi_monthly_start`
- `prev_bi_monthly_start_timestamp`

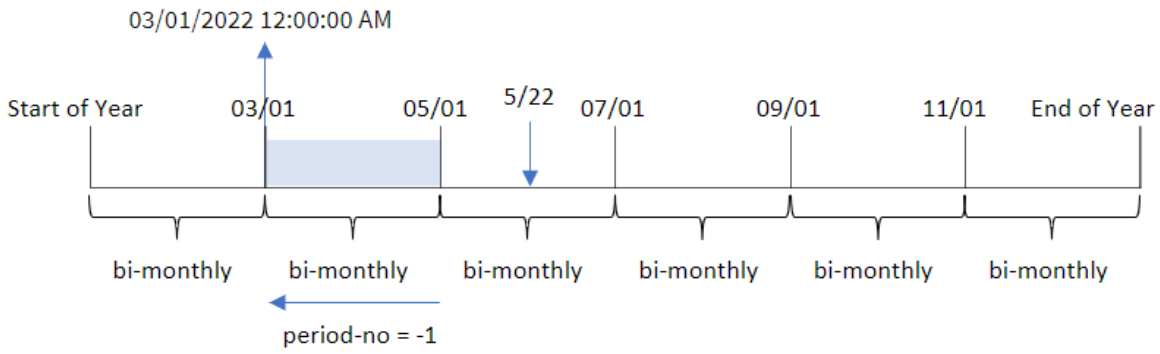
Ergebnistabelle

date	prev_bi_monthly_start	prev_bi_monthly_start_timestamp
2/19/2022	11/01/2021	11/1/2021 12:00:00 AM
3/7/2022	01/01/2022	1/1/2022 12:00:00 AM
3/30/2022	01/01/2022	1/1/2022 12:00:00 AM
4/5/2022	01/01/2022	1/1/2022 12:00:00 AM
4/16/2022	01/01/2022	1/1/2022 12:00:00 AM
5/1/2022	03/01/2022	3/1/2022 12:00:00 AM
5/7/2022	03/01/2022	3/1/2022 12:00:00 AM
5/22/2022	03/01/2022	3/1/2022 12:00:00 AM

date	prev_bi_monthly_start	prev_bi_monthly_start_timestamp
6/15/2022	03/01/2022	3/1/2022 12:00:00 AM
6/26/2022	03/01/2022	3/1/2022 12:00:00 AM
7/9/2022	05/01/2022	5/1/2022 12:00:00 AM
7/22/2022	05/01/2022	5/1/2022 12:00:00 AM
7/23/2022	05/01/2022	5/1/2022 12:00:00 AM
7/27/2022	05/01/2022	5/1/2022 12:00:00 AM
8/2/2022	05/01/2022	5/1/2022 12:00:00 AM
8/8/2022	05/01/2022	5/1/2022 12:00:00 AM
8/19/2022	05/01/2022	5/1/2022 12:00:00 AM
9/26/2022	07/01/2022	7/1/2022 12:00:00 AM
10/14/2022	07/01/2022	7/1/2022 12:00:00 AM
10/29/2022	07/01/2022	7/1/2022 12:00:00 AM

Da -1 als Argument `period_no` in der Funktion `monthsstart()` verwendet wird, gibt die Funktion, nachdem ein Jahr zunächst in Zweimonatssegmente unterteilt wurde, die erste Millisekunde des vorherigen Zweimonatssegments vor einer Transaktion zurück.

Diagramm der Funktion `monthsstart()`, Beispiel „period_no“



Transaktion 8195 fällt in das Segment zwischen dem Mai und Juni. Somit war das vorherige Zweimonatssegment der 1. März bis 30. April, und die Funktion gibt die erste Millisekunde dieses Segments zurück, den 1. März 2022 um 12:00:00 AM.

Beispiel 3 – first_month_of_year

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Derselbe Datensatz und dasselbe Szenario wie im ersten Beispiel.
- Es wird ein Feld `bi_monthly_start` erstellt, das Transaktionen in Zweimonatssegmenten gruppiert und den Startzeitstempel des Satzes für jede Transaktion zurückgibt.

In diesem Beispiel müssen wir aber zudem April als ersten Monat im Geschäftsjahr festlegen.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        monthsstart(2,date,0,4) as bi_monthly_start,
        timestamp(monthsstart(2,date,0,4)) as bi_monthly_start_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

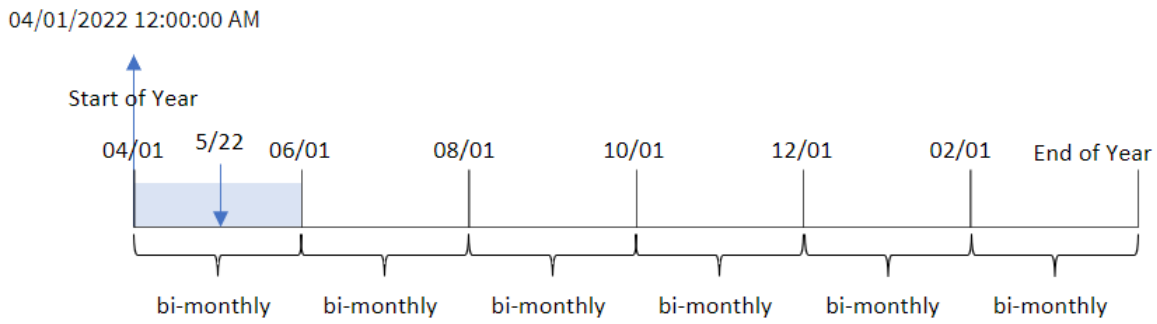
- `date`
- `bi_monthly_start`
- `bi_monthly_start_timestamp`

Ergebnistabelle

date	bi_monthly_start	bi_monthly_start_timestamp
2/19/2022	02/01/2022	2/1/2022 12:00:00 AM
3/7/2022	02/01/2022	2/1/2022 12:00:00 AM
3/30/2022	02/01/2022	2/1/2022 12:00:00 AM
4/5/2022	04/01/2022	4/1/2022 12:00:00 AM
4/16/2022	04/01/2022	4/1/2022 12:00:00 AM
5/1/2022	04/01/2022	4/1/2022 12:00:00 AM
5/7/2022	04/01/2022	4/1/2022 12:00:00 AM
5/22/2022	04/01/2022	4/1/2022 12:00:00 AM
6/15/2022	06/01/2022	6/1/2022 12:00:00 AM
6/26/2022	06/01/2022	6/1/2022 12:00:00 AM
7/9/2022	06/01/2022	6/1/2022 12:00:00 AM
7/22/2022	06/01/2022	6/1/2022 12:00:00 AM
7/23/2022	06/01/2022	6/1/2022 12:00:00 AM
7/27/2022	06/01/2022	6/1/2022 12:00:00 AM
8/2/2022	08/01/2022	8/1/2022 12:00:00 AM
8/8/2022	08/01/2022	8/1/2022 12:00:00 AM
8/19/2022	08/01/2022	8/1/2022 12:00:00 AM
9/26/2022	08/01/2022	8/1/2022 12:00:00 AM
10/14/2022	10/01/2022	10/1/2022 12:00:00 AM
10/29/2022	10/01/2022	10/1/2022 12:00:00 AM

Indem 4 als Argument `first_month_of_year` in der Funktion `monthsstart()` verwendet wird, beginnt die Funktion das Jahr am 1. April und unterteilt dann das Jahr in Zweimonatssegmente: Apr-May,Jun-Jul,Aug-Sep,Oct-Nov,Dec-Jan,Feb-Mar.

Diagramm der Funktion `monthsstart()`, Beispiel „first_month_of_year“



Transaktion 8195 fand am 22. Mai statt und fällt in das Segment zwischen dem 1. April und dem 31. Mai. Daher gibt die Funktion die erste Millisekunde dieses Segments zurück, den 1. April 2022 um 12:00:00 AM.

Beispiel 4 – Diagrammobjektbeispiel

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript verwendet den gleichen Datensatz und das gleiche Szenario wie das erste Beispiel.

In diesem Beispiel wird jedoch der unveränderte Datensatz in die Anwendung geladen. Die Berechnung, die Transaktionen in Zweimonatssegmenten gruppiert und den Startzeitstempel für den Satz für jede Transaktion zurückgibt, wird als Kennzahl in einem Diagrammobjekt der Anwendung erstellt.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8188,2/19/2022,37.23
8189,3/7/2022,17.17
8190,3/30/2022,88.27
8191,4/5/2022,57.42
8192,4/16/2022,53.80
8193,5/1/2022,82.06
8194,5/7/2022,40.39
8195,5/22/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
```

```
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu: `date`.

Erstellen Sie die folgenden Kennzahlen:

```
=monthsstart(2,date)
```

```
=timestamp(monthsstart(2,date))
```

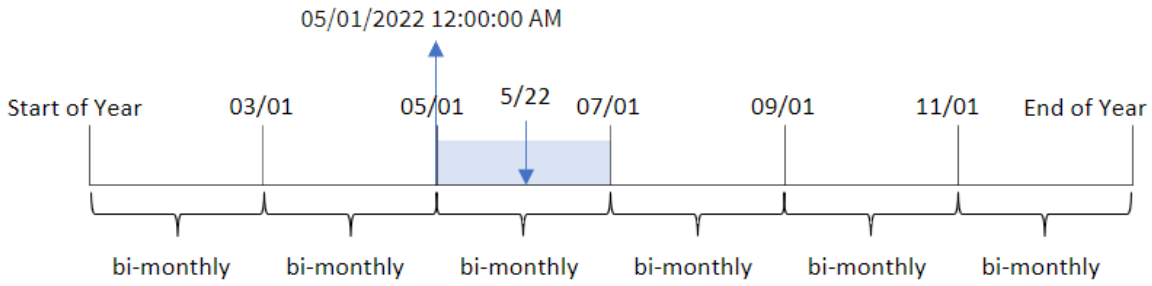
Mit diesen Berechnungen wird der Startzeitstempel des Zweimonatssegments abgerufen, in dem jede Transaktion stattfand.

Ergebnistabelle

date	=monthsstart(2,date)	=timestamp(monthsstart(2,date))
9/26/2022	09/01/2022	9/1/2022 12:00:00 AM
10/14/2022	09/01/2022	9/1/2022 12:00:00 AM
10/29/2022	09/01/2022	9/1/2022 12:00:00 AM
7/9/2022	07/01/2022	7/1/2022 12:00:00 AM
7/22/2022	07/01/2022	7/1/2022 12:00:00 AM
7/23/2022	07/01/2022	7/1/2022 12:00:00 AM
7/27/2022	07/01/2022	7/1/2022 12:00:00 AM
8/2/2022	07/01/2022	7/1/2022 12:00:00 AM
8/8/2022	07/01/2022	7/1/2022 12:00:00 AM
8/19/2022	07/01/2022	7/1/2022 12:00:00 AM
5/1/2022	05/01/2022	5/1/2022 12:00:00 AM
5/7/2022	05/01/2022	5/1/2022 12:00:00 AM
5/22/2022	05/01/2022	5/1/2022 12:00:00 AM
6/15/2022	05/01/2022	5/1/2022 12:00:00 AM
6/26/2022	05/01/2022	5/1/2022 12:00:00 AM

date	=monthsstart(2,date)	=timestamp(monthsstart(2,date))
3/7/2022	03/01/2022	3/1/2022 12:00:00 AM
3/30/2022	03/01/2022	3/1/2022 12:00:00 AM
4/5/2022	03/01/2022	3/1/2022 12:00:00 AM
4/16/2022	03/01/2022	3/1/2022 12:00:00 AM
2/19/2022	01/01/2022	1/1/2021 12:00:00 AM

Diagramm der Funktion `monthsstart()`, Diagrammobjektbeispiel



Transaktion 8195 fand am 22. Mai statt. Die Funktion `monthsstart()` unterteilt das Jahr zunächst in Zweimonatssegmente. Transaktion 8195 fällt in das Segment zwischen dem 1. Mai und dem 30. Juni. Daher gibt die Funktion die erste Millisekunde dieses Segments zurück, 05/01/2022 um 12:00:00 AM.

Beispiel 5 – Szenario

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz mit einer Reihe von Darlehenssalden wird in eine Tabelle namens `Loans` geladen.
- Die Daten bestehen aus der Darlehens-ID, dem Saldo zum Monatsbeginn und dem einfachen Zinssatz, der für jedes Darlehen pro Jahr berechnet wird.

Der Endbenutzer möchte ein Diagrammobjekt, das nach Darlehens-ID die aktuellen Zinsen anzeigt, die für jedes Darlehen im ausgewählten Zeitraum aufgelaufen sind. Das Geschäftsjahr beginnt im Januar.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Loans:
Load
```

```
*
Inline
[
loan_id,start_balance,rate
8188,$10000.00,0.024
8189,$15000.00,0.057
8190,$17500.00,0.024
8191,$21000.00,0.034
8192,$90000.00,0.084
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt.

Am Beginn des Ladeskripts wurde eine Variable (`vPeriod`) erstellt, die an die Variableneingabesteuerung gebunden wird. Dann wird die Variable als benutzerdefiniertes Objekt auf dem Arbeitsblatt konfiguriert.

Gehen Sie folgendermaßen vor:

1. Klicken Sie im Extras-Fenster auf **Benutzerdefinierte Objekte**.
2. Wählen Sie **Qlik Dashboard Bundle** aus und erstellen Sie ein Objekt **Variableneingabe**.
3. Geben Sie einen Titel für das Diagrammobjekt ein.
4. Wählen Sie unter **Variable** den Eintrag **vPeriod** als den Namen aus und legen Sie das Objekt so fest, dass es als **Dropdown** angezeigt wird.
5. Konfigurieren Sie unter **Werte** das Objekt für die Verwendung von dynamischen Werten. Geben Sie Folgendes ein:
   `='1~month|2~bi-month|3~quarter|4~tertial|6~half-year'`

Erstellen Sie dann die Ergebnistabelle.

Gehen Sie folgendermaßen vor:

1. Erstellen Sie eine neue Tabelle. Fügen Sie die folgenden Felder als Dimensionen hinzu:
   - `employee_id`
   - `employee_name`
2. Erstellen Sie eine Kennzahl, um die kumulierten Zinsen zu berechnen:
   `=start_balance*(rate*(today(1)-monthsstart($(vPeriod),today(1)))/365)`
3. Legen Sie das **Zahlenformat** der Kennzahl auf **Währung** fest. Klicken Sie auf ✔ **Bearbeitung fertig**. Jetzt können Sie das in der Tabelle angezeigte Datum ändern, indem Sie das Zeitsegment im Variablenobjekt anpassen.

Die Ergebnistabelle sieht wie folgt aus, wenn die Zeitraumoption `month` ausgewählt wird:

Ergebnistabelle

loan_id	start_balance	=start_balance*(rate*(today(1)-monthsstart($(vPeriod),today(1)))/365)
8188	$10000.00	$7.95

loan_id	start_balance	=start_balance*(rate*(today(1)-monthsstart($(vPeriod),today(1)))/365)
8189	$15000.00	$67.93
8190	$17500.00	$33.37
8191	$21000.00	$56.73
8192	$90000.00	$600.66

Die Funktion `monthsstart()` verwendet die Benutzereingabe als erstes Argument und das aktuelle Datum als zweites Argument. Dann gibt sie das Startdatum des vom Benutzer ausgewählten Zeitraums zurück. Die Formel zieht dieses Ergebnis vom aktuellen Datum ab und gibt die Anzahl der Tage zurück, die bisher in diesem Zeitraum verstrichen sind.

Dieser Wert wird dann mit dem Zinssatz multipliziert und durch 365 geteilt, um den effektiven Zinssatz für diesen Zeitraum zurückzugeben. Das Ergebnis wird dann mit dem Anfangssaldo des Darlehens multipliziert, was die Zinsen ergibt, die bislang in diesem Zeitraum aufgelaufen sind.

monthstart

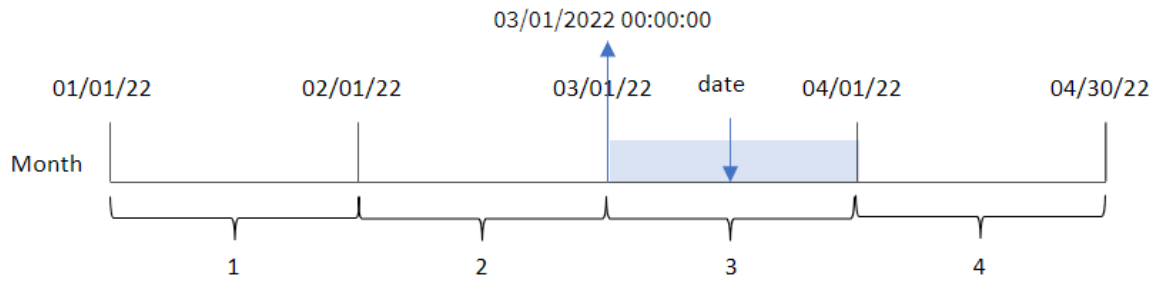
Diese Funktion liefert den Zeitstempel der ersten Millisekunde des ersten Tages des Monats, in dem **date** liegt. Das Ergebnis wird entsprechend dem im Skript definierten **DateFormat** formatiert.

Syntax:

```
MonthStart(date[, period_no])
```

Rückgabe Datentyp: dual

Diagramm der Funktion `monthstart()`



Die Funktion `monthstart()` bestimmt, in welchen Monat das Datum fällt. Sie gibt dann einen Zeitstempel im Datumsformat für die erste Millisekunde dieses Monats zurück.

Argumente

Argument	Beschreibung
date	Datum oder Zeitstempel für die Evaluierung.

Argument	Beschreibung
period_no	**period_no** ist eine ganze Zahl, die beim Weglassen von 0 den Monat liefert, der **date** enthält. Negative Werte von **period_no** stehen für vorangehende, positive Werte für nachfolgende Monate.

Verwendung

Die Funktion `monthstart()` wird in der Regel als Teil einer Formel verwendet, wenn in der Berechnung der Teil des Monats verwendet werden soll, der bereits verstrichen ist. Zum Beispiel kann sie verwendet werden, um die Zinsen zu berechnen, die in einem Monat bis zu einem bestimmten Datum aufgelaufen sind.

Funktionsbeispiele

Beispiel	Ergebnis
`monthstart('10/19/2001')`	Gibt 10/01/2001 zurück.
`monthstart('10/19/2001', -1)`	Gibt 09/01/2001 zurück.

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Beispiel 1 – keine zusätzlichen Argumente

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz enthält eine Reihe von Transaktionen für 2022 und wird in eine Tabelle namens `Transactions` geladen.
- Das Datumsfeld wird im Format der Systemvariablen `DateFormat` (MM/TT/JJJJ) bereitgestellt.

- Es wird ein Feld `start_of_month` erstellt, das einen Zeitstempel für den Start des Monats zurückgibt, in dem die Transaktionen stattfanden.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        monthstart(date) as start_of_month,
        timestamp(monthstart(date)) as start_of_month_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `start_of_month`
- `start_of_month_timestamp`

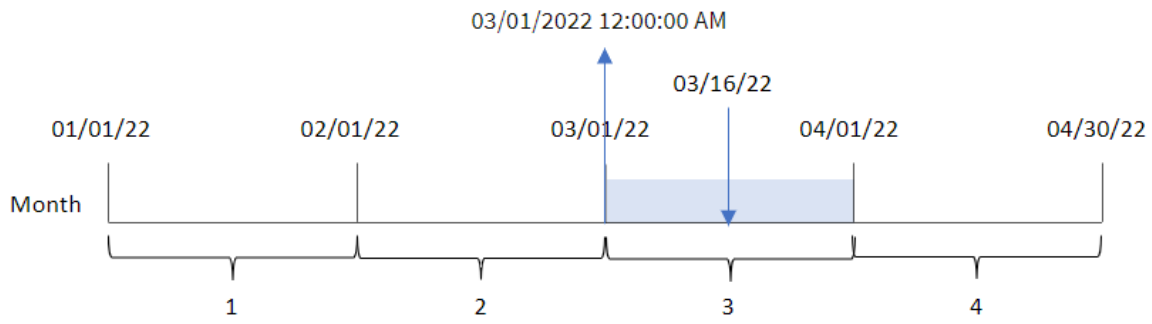
Ergebnistabelle

date	start_of_month	start_of_month_timestamp
1/7/2022	01/01/2022	1/1/2022 12:00:00 AM
1/19/2022	01/01/2022	1/1/2022 12:00:00 AM
2/5/2022	02/01/2022	2/1/2022 12:00:00 AM
2/28/2022	02/01/2022	2/1/2022 12:00:00 AM
3/16/2022	03/01/2022	3/1/2022 12:00:00 AM
4/1/2022	04/01/2022	4/1/2022 12:00:00 AM
5/7/2022	05/01/2022	5/1/2022 12:00:00 AM
5/16/2022	05/01/2022	5/1/2022 12:00:00 AM
6/15/2022	06/01/2022	6/1/2022 12:00:00 AM
6/26/2022	07/01/2022	6/1/2022 12:00:00 AM
7/9/2022	07/01/2022	7/1/2022 12:00:00 AM
7/22/2022	07/01/2022	7/1/2022 12:00:00 AM
7/23/2022	07/01/2022	7/1/2022 12:00:00 AM
7/27/2022	07/01/2022	7/1/2022 12:00:00 AM
8/2/2022	08/01/2022	8/1/2022 12:00:00 AM
8/8/2022	08/01/2022	8/1/2022 12:00:00 AM
8/19/2022	08/01/2022	8/1/2022 12:00:00 AM
9/26/2022	09/01/2022	9/1/2022 12:00:00 AM
10/14/2022	10/01/2022	10/1/2022 12:00:00 AM
10/29/2022	10/01/2022	10/1/2022 12:00:00 AM

Das Feld `start_of_month` wird im vorangehenden load-Befehl erstellt, indem die Funktion `monthstart()` verwendet und das Datumsfeld als Argument der Funktion übergeben wird.

Die Funktion `monthstart()` identifiziert, in welchen Monat der Datumswert fällt, und gibt einen Zeitstempel für die erste Millisekunde dieses Monats zurück.

Diagramm der Funktion `monthstart()`, Beispiel ohne zusätzliche Argumente



Transaktion 8192 fand am 16. März statt. Die Funktion `monthstart()` gibt die erste Millisekunde dieses Monats zurück, also den 1. März um 12:00:00 AM.

Beispiel 2 – period_no

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Derselbe Datensatz und dasselbe Szenario wie im ersten Beispiel.
- Es wird ein Feld `previous_month_start` erstellt, das den Zeitstempel für den Start des Monats vor dem Monat zurückgibt, in dem die Transaktion stattfand.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        monthstart(date,-1) as previous_month_start,
        timestamp(monthstart(date,-1)) as previous_month_start_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
```

```
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `previous_month_start`
- `previous_month_start_timestamp`

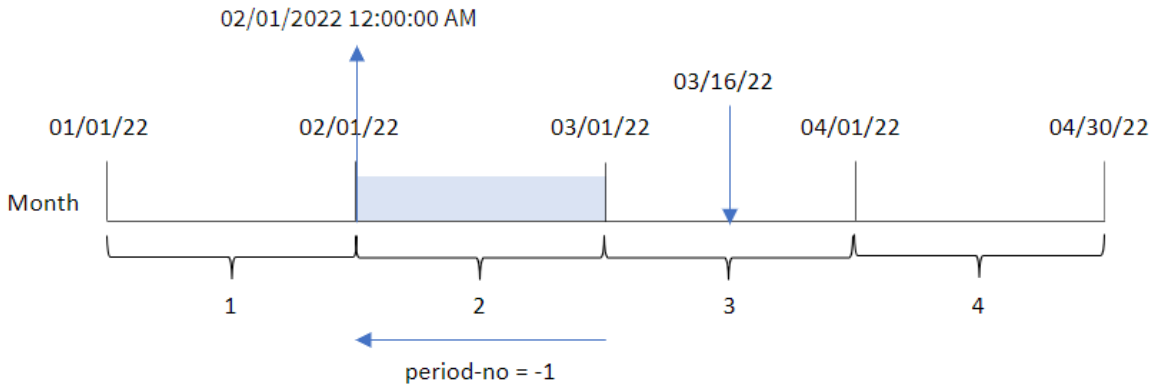
Ergebnistabelle

date	previous_month_start	previous_month_start_timestamp
1/7/2022	12/01/2021	12/1/2021 12:00:00 AM
1/19/2022	12/01/2021	12/1/2021 12:00:00 AM
2/5/2022	01/01/2022	1/1/2022 12:00:00 AM
2/28/2022	01/01/2022	1/1/2022 12:00:00 AM
3/16/2022	02/01/2022	2/1/2022 12:00:00 AM
4/1/2022	03/01/2022	3/1/2022 12:00:00 AM
5/7/2022	04/01/2022	4/1/2022 12:00:00 AM
5/16/2022	04/01/2022	4/1/2022 12:00:00 AM
6/15/2022	05/01/2022	5/1/2022 12:00:00 AM
6/26/2022	05/01/2022	5/1/2022 12:00:00 AM
7/9/2022	06/01/2022	6/1/2022 12:00:00 AM
7/22/2022	06/01/2022	6/1/2022 12:00:00 AM
7/23/2022	06/01/2022	6/1/2022 12:00:00 AM
7/27/2022	06/01/2022	6/1/2022 12:00:00 AM

date	previous_month_start	previous_month_start_timestamp
8/2/2022	07/01/2022	7/1/2022 12:00:00 AM
8/8/2022	07/01/2022	7/1/2022 12:00:00 AM
8/19/2022	07/01/2022	7/1/2022 12:00:00 AM
9/26/2022	08/01/2022	8/1/2022 12:00:00 AM
10/14/2022	09/01/2022	9/1/2022 12:00:00 AM
10/29/2022	09/01/2022	9/1/2022 12:00:00 AM

Da in diesem Fall eine `period_no` von -1 als Versatzargument in der Funktion `monthstart()` verwendet wurde, identifiziert die Funktion zuerst den Monat, in dem die Transaktionen stattfanden. Dann geht sie einen Monat zurück und identifiziert die erste Millisekunde dieses Monats.

Diagramm der Funktion `monthstart()`, Beispiel „period_no“



Transaktion 8192 fand am 16. März statt. Die Funktion `monthstart()` identifiziert, dass der Monat vor dem Transaktionsdatum der Februar war. Dann wird die erste Millisekunde dieses Monats zurückgegeben, der 1. Februar um 12:00:00 AM.

Beispiel 3 – Diagrammobjektbeispiel

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript verwendet den gleichen Datensatz und das gleiche Szenario wie das erste Beispiel.

In diesem Beispiel wird jedoch der unveränderte Datensatz in die Anwendung geladen. Die Berechnung, die einen Zeitstempel für den Start des Monats zurückgibt, in dem die Transaktionen stattfanden, wird als Kennzahl in einem Diagrammobjekt der Anwendung erstellt.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu: `date`.

Um das Startdatum des Monats zu berechnen, in dem eine Transaktion stattfindet, erstellen Sie die folgenden Kennzahlen:

- `=monthstart(date)`
- `=timestamp(monthstart(date))`

Ergebnistabelle

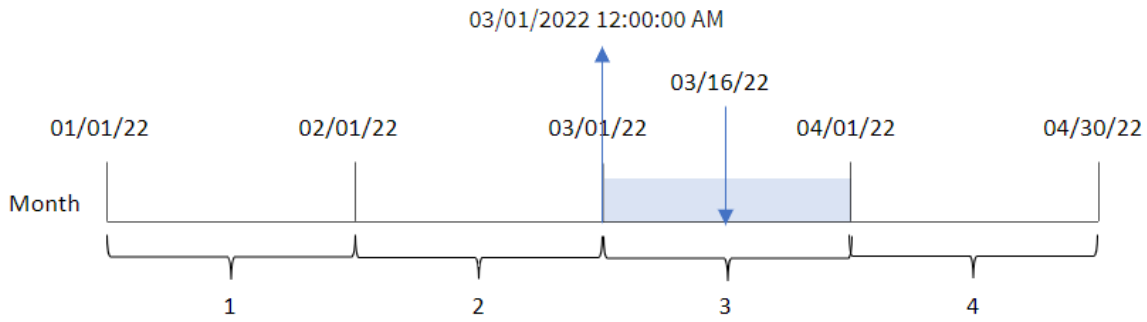
date	=monthstart(date)	=timestamp(monthstart(date))
10/14/2022	10/01/2022	10/1/2022 12:00:00 AM
10/29/2022	10/01/2022	10/1/2022 12:00:00 AM
9/26/2022	09/01/2022	9/1/2022 12:00:00 AM

date	=monthstart(date)	=timestamp(monthstart(date))
8/2/2022	08/01/2022	8/1/2022 12:00:00 AM
8/8/2022	08/01/2022	8/1/2022 12:00:00 AM
8/19/2022	08/01/2022	8/1/2022 12:00:00 AM
7/9/2022	07/01/2022	7/1/2022 12:00:00 AM
7/22/2022	07/01/2022	7/1/2022 12:00:00 AM
7/23/2022	07/01/2022	7/1/2022 12:00:00 AM
7/27/2022	07/01/2022	7/1/2022 12:00:00 AM
6/15/2022	06/01/2022	6/1/2022 12:00:00 AM
6/26/2022	06/01/2022	6/1/2022 12:00:00 AM
5/7/2022	05/01/2022	5/1/2022 12:00:00 AM
5/16/2022	05/01/2022	5/1/2022 12:00:00 AM
4/1/2022	04/01/2022	4/1/2022 12:00:00 AM
3/16/2022	03/01/2022	3/1/2022 12:00:00 AM
2/5/2022	02/01/2022	2/1/2022 12:00:00 AM
2/28/2022	02/01/2022	2/1/2022 12:00:00 AM
1/7/2022	01/01/2022	1/1/2022 12:00:00 AM
1/19/2022	01/01/2022	1/1/2022 12:00:00 AM

Die Kennzahl `start_of_month` wird im Diagrammobjekt erstellt, indem die Funktion `monthstart()` verwendet und das Datumsfeld als Argument der Funktion übergeben wird.

Die Funktion `monthstart()` identifiziert, in welchen Monat der Datumswert fällt, und gibt einen Zeitstempel für die erste Millisekunde dieses Monats zurück.

Diagramm der Funktion `monthstart()`, Diagrammobjektbeispiel



Transaktion 8192 fand am 16. März statt. Die Funktion `monthstart()` identifiziert, dass die Transaktion im März stattfand, und gibt die erste Millisekunde dieses Monats zurück, also den 1. März um 12:00:00 AM.

Beispiel 4 – Szenario

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz mit einer Reihe von Darlehenssalden wird in eine Tabelle namens `Loans` geladen.
- Die Daten bestehen aus der Darlehens-ID, dem Saldo zum Monatsbeginn und dem einfachen Zinssatz, der für jedes Darlehen pro Jahr berechnet wird.

Der Endbenutzer möchte ein Diagrammobjekt, das nach Darlehens-ID die aktuellen Zinsen anzeigt, die für jedes Darlehen im Monat bis dato aufgelaufen sind.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Loans:
Load
*
Inline
[
loan_id,start_balance,rate
8188,$10000.00,0.024
8189,$15000.00,0.057
8190,$17500.00,0.024
8191,$21000.00,0.034
8192,$90000.00,0.084
];
```

Ergebnisse

Gehen Sie folgendermaßen vor:

1. Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:
    - `loan_id`
    - `start_balance`
2. Erstellen Sie dann eine Kennzahl, um die kumulierten Zinsen zu berechnen: `=start_balance*(rate*(today(1)-monthstart(today(1)))/365)`
3. Legen Sie das **Zahlenformat** der Kennzahl auf **Währung** fest.

Ergebnistabelle

loan_id	start_balance	=start_balance*(rate*(today(1)-monthstart(today(1)))/365)
8188	$10000.00	$16.44
8189	$15000.00	$58.56
8190	$17500.00	$28.77
8191	$21000.00	$48.90
8192	$90000.00	$517.81

Die Funktion `monthstart()` verwendet das aktuelle Datum als einziges Argument und gibt das Startdatum des aktuellen Monats zurück. Die Formel zieht dieses Ergebnis vom aktuellen Datum ab und gibt die Anzahl der Tage zurück, die bisher im Monat verstrichen sind.

Dieser Wert wird dann mit dem Zinssatz multipliziert und durch 365 geteilt, um den effektiven Zinssatz für diesen Zeitraum zurückzugeben. Das Ergebnis wird dann mit dem Anfangssaldo des Darlehens multipliziert, was die Zinsen ergibt, die bislang in diesem Monat aufgelaufen sind.

networkdays

Die Funktion **networkdays** liefert die Zahl der Arbeitstage (Montag bis Freitag) zwischen **start_date** und **end_date**, unter Berücksichtigung eventueller Feiertage unter **holiday**.

Syntax:

```
networkdays (start_date, end_date [, holiday])
```

Rückgabe Datentyp: ganze Zahl

Ein Kalenderdiagramm zeigt den Datumsbereich, der von der Funktion „networkdays“ zurückgegeben wird.

Sun	Mon	Tue	Wed	Thu	Fri	Sat
	1	2	3	4	5	6
7	8	9	10 start_date	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26 end_date	27
28	29	30	31			

Die Funktion `networkdays` weist die folgenden Beschränkungen auf:

- Es gibt keine Methode zum Ändern von Werktagen. Somit gibt es keine Möglichkeit zum Ändern der Funktion für Regionen oder Situationen, in denen anders als von Montag bis Freitag gearbeitet wird.
- Der Parameter `holiday` muss eine String-Konstante sein. Formeln werden nicht akzeptiert.

Argumente

Argument	Beschreibung
start_date	Das auszuwertende Startdatum.
end_date	Das auszuwertende Enddatum.
holiday	Feiertagszeiträume, die von den Arbeitstagen auszuschließen sind. Ein Feiertag wird als ein Datum mit Zeichenfolgenkonstante angegeben. Sie können mehrere Feiertagstermine getrennt durch Kommas festlegen. **Beispiel:** `'12/25/2013', '12/26/2013', '12/31/2013', '01/01/2014'`

Verwendung

Die Funktion `networkdays()` wird in der Regel als Teil einer Formel verwendet, wenn in der Berechnung die Anzahl der Arbeitswochentage verwendet werden soll, die zwischen zwei Datumswerten auftreten. Beispielsweise möchte ein Benutzer den Gesamtlohn berechnen, der von einem Mitarbeiter mit einem Vertrag mit Lohnsteuerabzug verdient wird.

Funktionsbeispiele

Beispiel	Ergebnis
`networkdays ('12/19/2013', '01/07/2014')`	Gibt 14 zurück. In diesem Beispiel werden Feiertage nicht berücksichtigt.
`networkdays ('12/19/2013', '01/07/2014', '12/25/2013', '12/26/2013')`	Gibt 12 zurück. In diesem Beispiel wird der Feiertagszeitraum vom 12/25/2013 bis zum 12/26/2013 berücksichtigt.
`networkdays ('12/19/2013', '01/07/2014', '12/25/2013', '12/26/2013', '12/31/2013', '01/01/2014')`	Gibt 10 zurück. Dieses Beispiel berücksichtigt zwei Feiertagszeiträume.

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Beispiel 1 – einfaches Beispiel

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz enthält Projekt-IDs sowie Start- und Enddatum der Projekte. Diese Informationen werden in eine Tabelle namens `Projects` geladen.
- Das Datumsfeld wird im Format der Systemvariablen `DateFormat` (MM/TT/JJJJ) bereitgestellt.
- Es wird ein zusätzliches Feld `net_work_days` erstellt, um die Anzahl der Werktage für jedes Projekt zu berechnen.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';
```

```
Projects:
    Load
        *,
        networkdays(start_date,end_date) as net_work_days
        ;
Load
id,
start_date,
end_date
Inline
[
id,start_date,end_date
1,01/01/2022,01/18/2022
2,02/10/2022,02/17/2022
3,05/17/2022,07/05/2022
4,06/01/2022,06/12/2022
5,08/10/2022,08/26/2022
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `id`
- `start_date`
- `end_date`
- `net_work_days`

Ergebnistabelle

id	start_date	end_date	net_work_days
1	01/01/2022	01/18/2022	12
2	02/10/2022	02/17/2022	6
3	05/17/2022	07/05/2022	36
4	06/01/2022	06/12/2022	8
5	08/10/2022	08/26/2022	13

Da keine geplanten Feiertage vorhanden sind (diese wären im dritten Argument der Funktion `networkdays()` enthalten gewesen), zieht die Funktion das `start_date` vom `end_date` sowie alle Wochenenden ab, um die Anzahl der Werktage zwischen den beiden Datumswerten zu berechnen.

Das Kalenderdiagramm hebt die Werktage für Projekt 5 hervor (keine Feiertage)

Sun	Mon	Tue	Wed	Thu	Fri	Sat
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

Der obige Kalender zeigt ein visuelles Schema des Projekts mit der `id` 5. Projekt 5 beginnt am Mittwoch, den 10. August 2022 und endet am 26. August 2022. Alle Samstage und Sonntage werden ignoriert, und zwischen diesen Datumswerten (beide einschließlich) liegen 13 Werktage.

Beispiel 2 – Ein Feiertag

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Derselbe Datensatz und dasselbe Szenario wie im vorigen Beispiel.
- Das Datumsfeld wird im Format der Systemvariablen `DateFormat` (MM/TT/JJJJ) bereitgestellt.
- Es wird ein zusätzliches Feld `net_work_days` erstellt, um die Anzahl der Werktage für jedes Projekt zu berechnen.

In diesem Beispiel ist ein eintägiger Feiertag am 19. August 2022 geplant.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Projects:
```

```
    Load
        *,
        networkdays(start_date,end_date,'08/19/2022') as net_work_days
        ;
Load
id,
start_date,
end_date
Inline
[
id,start_date,end_date
1,01/01/2022,01/18/2022
2,02/10/2022,02/17/2022
3,05/17/2022,07/05/2022
4,06/01/2022,06/12/2022
5,08/10/2022,08/26/2022
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `id`
- `start_date`
- `end_date`
- `net_work_days`

Ergebnistabelle

id	start_date	end_date	net_work_days
1	01/01/2022	01/18/2022	12
2	02/10/2022	02/17/2022	6
3	05/17/2022	07/05/2022	36
4	06/01/2022	06/12/2022	8
5	08/10/2022	08/26/2022	12

Der eintägige geplante Feiertag wird als drittes Argument in die Funktion `networkdays()` eingegeben.

Das Kalenderdiagramm hebt die Werktage für Projekt 5 hervor (eintägiger Feiertag)

Sun	Mon	Tue	Wed	Thu	Fri	Sat
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19 Holiday	20
21	22	23	24	25	26	27
28	29	30	31			

Der obige Kalender zeigt ein visuelles Schema von Projekt 5. Er wurde angepasst, um den Feiertag zu berücksichtigen. Dieser Feiertag liegt in Projekt 5 am Freitag, den 19. August 2022. Infolgedessen reduziert sich der Gesamtwert von `net_work_days` für Projekt 5 um einen Tag, von 13 auf 12 Tage.

Beispiel 3 – Mehrere Feiertage

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Derselbe Datensatz und dasselbe Szenario wie im ersten Beispiel.
- Das Datumsfeld wird im Format der Systemvariablen `DateFormat` (MM/TT/JJJJ) bereitgestellt.
- Es wird ein zusätzliches Feld `net_work_days` erstellt, um die Anzahl der Werktage für jedes Projekt zu berechnen.

In diesem Beispiel sind jedoch vier Feiertage vom 18. August bis zum 21. August 2022 geplant.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Projects:
    Load
        *,
        networkdays(start_date,end_date,'08/18/2022','08/19/2022','08/20/2022','08/21/2022')
as net_work_days
        ;
Load
id,
start_date,
end_date
Inline
[
id,start_date,end_date
1,01/01/2022,01/18/2022
2,02/10/2022,02/17/2022
3,05/17/2022,07/05/2022
4,06/01/2022,06/12/2022
5,08/10/2022,08/26/2022
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `id`
- `start_date`
- `end_date`
- `net_work_days`

Ergebnistabelle

id	start_date	end_date	net_work_days
1	01/01/2022	01/18/2022	12
2	02/10/2022	02/17/2022	6
3	05/17/2022	07/05/2022	36
4	06/01/2022	06/12/2022	8
5	08/10/2022	08/26/2022	11

Die vier geplanten Feiertage werden als kommagetrennte Liste ab dem dritten Argument in der Funktion `networkdays()` eingegeben.

Das Kalenderdiagramm hebt die Werktage für Projekt 5 hervor (mehrere Feiertage)

Sun	Mon	Tue	Wed	Thu	Fri	Sat
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18 Holiday	19 Holiday	20
21	22	23	24	25	26	27
28	29	30	31			

Der obige Kalender zeigt ein visuelles Schema von Projekt 5 mit der Anpassung, um diese Feiertage zu berücksichtigen. Dieser Zeitraum mit geplanten Feiertagen lag in Projekt 5, wobei zwei der Tage auf einem Donnerstag und Freitag lagen. Infolgedessen reduziert sich der Gesamtwert von `net_work_days` für Projekt 5 um zwei Tage, von 13 auf 11 Tage.

Beispiel 4 – Ein Feiertag

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Derselbe Datensatz und dasselbe Szenario wie im ersten Beispiel.
- Das Datumsfeld wird im Format der Systemvariablen `DateFormat` (MM/TT/JJJJ) bereitgestellt.

Es ist ein eintägiger Feiertag am 19. August 2022 geplant.

In diesem Beispiel wird jedoch der unveränderte Datensatz in die Anwendung geladen. Das Feld net_work_days wird als Kennzahl in einem Diagrammobjekt berechnet.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Projects:
Load
id,
start_date,
end_date
Inline
[
id,start_date,end_date
1,01/01/2022,01/18/2022
2,02/10/2022,02/17/2022
3,05/17/2022,07/05/2022
4,06/01/2022,06/12/2022
5,08/10/2022,08/26/2022
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- id
- start_date
- end_date

Erstellen Sie die folgende Kennzahl:

```
= networkdays(start_date,end_date,'08/19/2022')
```

Ergebnistabelle

id	start_date	end_date	net_work_days
1	01/01/2022	01/18/2022	12
2	02/10/2022	02/17/2022	6
3	05/17/2022	07/05/2022	36
4	06/01/2022	06/12/2022	8
5	08/10/2022	08/26/2022	12

Der eintägige geplante Feiertag wird als drittes Argument in die Funktion networkdays() eingegeben.

Das Kalenderdiagramm zeigt die Nettowerktage mit einem eintägigen Feiertag (Diagrammobjekt)

Sun	Mon	Tue	Wed	Thu	Fri	Sat
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19 Holiday	20
21	22	23	24	25	26	27
28	29	30	31			

Der obige Kalender zeigt ein visuelles Schema von Projekt 5. Er wurde angepasst, um den Feiertag zu berücksichtigen. Dieser Feiertag liegt in Projekt 5 am Freitag, den 19. August 2022. Infolgedessen reduziert sich der Gesamtwert von `net_work_days` für Projekt 5 um einen Tag, von 13 auf 12 Tage.

now

Diese Funktion gibt einen Zeitstempel der aktuellen Uhrzeit zurück. Die Funktion gibt Werte im Systemvariablenformat **TimeStamp** zurück. Der Standardwert von **timer_mode** ist 1.

Syntax:

```
now([ timer_mode])
```

Rückgabe Datentyp: dual

Die Funktion `now()` kann entweder im Ladeskript oder in Diagrammobjekten verwendet werden.

Argumente

Argument	Beschreibung
timer_mode	Kann folgende Werte haben: 0 (Uhrzeit des letzten abgeschlossenen Datenladevorgangs) 1 (Uhrzeit des Funktionsaufrufs) 2 (Uhrzeit beim Öffnen der App) *Wenn Sie die Funktion im Datenladeskript verwenden, ergibt **timer_mode=0** die Uhrzeit des letzten abgeschlossenen Datenladevorgangs, während **timer_mode=1** die Zeit des Funktionsaufrufs zum Laden der aktuellen Daten angibt.*

Die Funktion `now()` hat eine hochperformante Auswirkung, was zu Scrollproblemen führen könnte, wenn die Funktion innerhalb der Formeln von Tabellen verwendet wird. Sofern deren Verwendung nicht absolut notwendig ist, empfehlen wir, stattdessen die Funktion `today()` zu verwenden. Wenn die Verwendung von `now()` in einem Layout erforderlich ist, empfehlen wir, sofern möglich, die nicht standardmäßigen Einstellungen `now(0)` oder `now(2)`, da hierfür keine ständigen Neuberechnungen nötig sind.

Verwendung

Die Funktion `now()` wird häufig als Komponente innerhalb einer Formel verwendet. Beispielsweise kann sie verwendet werden, um die verbleibende Zeit im Lebenszyklus eines Produkts zu berechnen. Die Funktion `now()` kann anstelle der Funktion `today()` verwendet werden, wenn in der Formel ein Bruchteil eines Tages verwendet werden muss.

Die folgende Tabelle erläutert das Ergebnis, das von der Funktion `now()` bei verschiedenen Werten für das Argument `timer_mode` zurückgegeben wird:

Funktionsbeispiele

timer_mode value	Ergebnis bei Verwendung im Ladeskript	Ergebnis bei Verwendung im Diagrammobjekt
0	Gibt einen Zeitstempel im Format der Systemvariablen `TimeStamp` für den letzten erfolgreichen Datenladevorgang vor dem neuesten Datenladevorgang zurück.	Gibt einen Zeitstempel im Format der Systemvariablen `TimeStamp` für den neuesten Datenladevorgang zurück.
1	Gibt einen Zeitstempel im Format der Systemvariablen `TimeStamp` für den neuesten Datenladevorgang zurück.	Gibt einen Zeitstempel im Format der Systemvariablen `TimeStamp` für den Funktionsaufruf zurück.

timer_mode value	Ergebnis bei Verwendung im Ladeskript	Ergebnis bei Verwendung im Diagrammobjekt
2	Gibt einen Zeitstempel im Format der Systemvariablen `TimeStamp` für den Zeitpunkt zurück, zu dem die Sitzung des Benutzers in der Anwendung begann. Dieser wird erst aktualisiert, wenn der Benutzer das Skript neu lädt.	Gibt den Zeitstempel im Format der Systemvariablen `TimeStamp` für den Zeitpunkt zurück, zu dem die Sitzung des Benutzers in der Anwendung begann. Dieser wird aktualisiert, wenn eine neue Sitzung beginnt oder wenn die Daten in der Anwendung neu geladen werden.

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Beispiel 1 – Generierung von Objekten anhand des Ladeskripts

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

In diesem Beispiel werden drei Variablen mit der Funktion `now()` erstellt. Jede Variable verwendet eine der Optionen `timer_mode`, um deren Auswirkung zu demonstrieren.

Damit der Zweck der Variablen demonstriert wird, laden Sie das Skript und laden Sie es nach einer kurzen Wartezeit ein zweites Mal. Das führt dazu, dass die Variablen `now(0)` und `now(1)` unterschiedliche Werte zeigen und damit ihren Zweck demonstrieren.

Ladeskript

```
LET vPreviousDataLoad = now(0);
LET vCurrentDataLoad = now(1);
LET vApplicationOpened = now(2);
```

Ergebnisse

Erstellen Sie drei Textfelder anhand der Anleitung unten, sobald die Daten ein zweites Mal geladen wurden.

Erstellen Sie zunächst ein Textfeld für zuvor geladene Daten.

Gehen Sie folgendermaßen vor:

1. Erstellen Sie ein Textfeld mithilfe des Diagrammobjekts **Text und Bild**.
2. Fügen Sie dem Objekt die folgende Kennzahl hinzu:
   `=vPreviousDataLoad`
3. Wählen Sie unter **Darstellung** die Option **Show titles** aus und fügen Sie den Titel „Zeit des vorherigen Neuladevorgangs“ zum Objekt hinzu.

Erstellen Sie als Nächstes ein Textfeld für Daten, die derzeit geladen werden.

Gehen Sie folgendermaßen vor:

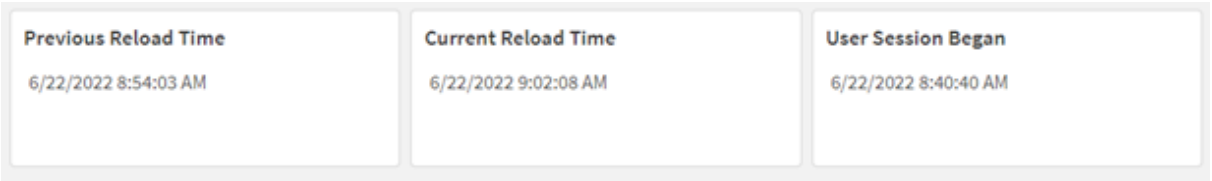
1. Erstellen Sie ein Textfeld mithilfe des Diagrammobjekts **Text und Bild**.
2. Fügen Sie dem Objekt die folgende Kennzahl hinzu:
   `=vCurrentDataLoad`
3. Wählen Sie unter **Darstellung** die Option **Show titles** aus und fügen Sie den Titel „Zeit des aktuellen Neuladevorgangs“ zum Objekt hinzu.

Erstellen Sie ein letztes Textfeld, das anzeigt, wann die Sitzung des Benutzers in der Anwendung gestartet wurde.

Gehen Sie folgendermaßen vor:

1. Erstellen Sie ein Textfeld mithilfe des Diagrammobjekts **Text und Bild**.
2. Fügen Sie dem Objekt die folgende Kennzahl hinzu:
   `=vApplicationOpened`
3. Wählen Sie unter **Darstellung** die Option **Show titles** aus und fügen Sie den Titel „Start der Benutzersitzung“ zum Objekt hinzu.

Variablen des Ladeskripts now()



Die obige Abbildung zeigt Beispielwerte für jede der erstellten Variablen. Die Werte können beispielsweise wie folgt aussehen:

- Vorherige Ladezeit: 6/22/2022 8:54:03 AM
- Aktuelle Ladezeit: 6/22/2022 9:02:08 AM
- Beginn der Benutzersitzung: 6/22/2022 8:40:40 AM

Beispiel 2 – Generierung von Objekten ohne Ladeskript

Ladeskript und Diagrammformel

Übersicht

In diesem Beispiel erstellen Sie drei Diagrammobjekte anhand der Funktion `now()`, ohne Variablen oder Daten in die Anwendung zu laden. Jedes Diagrammobjekt verwendet eine der Optionen für `timer_mode`, um deren Auswirkung zu demonstrieren.

Für dieses Beispiel gibt es kein Ladeskript.

Gehen Sie folgendermaßen vor:

1. Öffnen Sie den Dateneditor.
2. Ändern Sie das vorhandene Ladeskript nicht und klicken Sie auf **Daten laden**.
3. Laden Sie das Skript nach einer kurzen Wartezeit ein zweites Mal.

Ergebnisse

Nachdem die Daten ein zweites Mal geladen wurden, erstellen Sie drei Textfelder.

Erstellen Sie zuerst ein Textfeld für den neuesten Datenladevorgang.

Gehen Sie folgendermaßen vor:

1. Erstellen Sie ein Textfeld mithilfe des Diagrammobjekts **Text und Bild**.
2. Fügen Sie die folgende Kennzahl hinzu.
   `=now(0)`
3. Wählen Sie unter **Darstellung** die Option **Titel anzeigen** aus und fügen Sie den Titel „Neuester Datenladevorgang“ zum Objekt hinzu.

Erstellen Sie dann ein Textfeld, das die aktuelle Uhrzeit anzeigt.

Gehen Sie folgendermaßen vor:

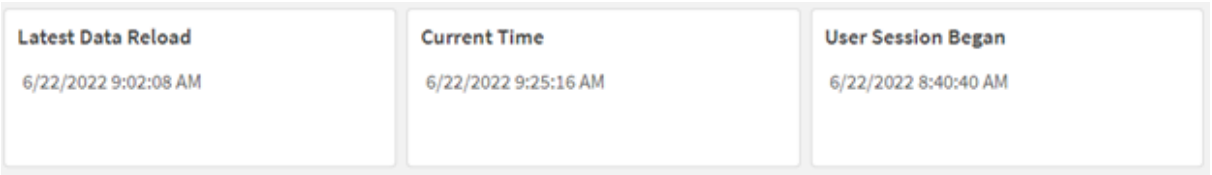
1. Erstellen Sie ein Textfeld mithilfe des Diagrammobjekts **Text und Bild**.
2. Fügen Sie die folgende Kennzahl hinzu:
   `=now(1)`
3. Wählen Sie unter **Darstellung** die Option**Titel anzeigen** aus und fügen Sie den Titel „Aktuelle Uhrzeit“ zum Objekt hinzu.

Erstellen Sie ein letztes Textfeld, das anzeigt, wann die Sitzung des Benutzers in der Anwendung gestartet wurde.

Gehen Sie folgendermaßen vor:

1. Erstellen Sie ein Textfeld mithilfe des Diagrammobjekts **Text und Bild**.
2. Fügen Sie die folgende Kennzahl hinzu:
   `=now(2)`
3. Wählen Sie unter **Darstellung** die Option **Titel anzeigen** aus und fügen Sie den Titel „Beginn der Benutzersitzung“ zum Objekt hinzu.

Beispiele für das Diagrammobjekt now()



Die obige Abbildung zeigt Beispielwerte für jedes der erstellten Objekte. Die Werte können beispielsweise wie folgt aussehen:

- Neuester Datenladevorgang: 6/22/2022 9:02:08 AM
- Aktuelle Uhrzeit: 6/22/2022 9:25:16 AM
- Beginn der Benutzersitzung: 6/22/2022 8:40:40 AM

Das Diagrammobjekt „Neuester Datenladevorgang“ verwendet einen Wert für `timer_mode` von 0. Damit wird ein Zeitstempel für die Uhrzeit zurückgegeben, zu der die Daten zum letzten Mal erfolgreich geladen wurden.

Das Diagrammobjekt „Aktuelle Uhrzeit“ verwendet einen Wert für `timer_mode` von 1. Damit wird die aktuelle Uhrzeit der Systemuhr zurückgegeben. Wenn das Arbeitsblatt oder Objekt aktualisiert wird, wird dieser Wert aktualisiert.

Das Diagrammobjekt „Beginn der Benutzersitzung“ verwendet einen Wert für `timer_mode` von 2. Damit wird der Zeitstempel des Zeitpunkts zurückgegeben, zu dem die Anwendung geöffnet wurde und die Benutzersitzung begann.

Beispiel 3 – Szenario

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz mit dem Verzeichnis für das Schürfen einer Kryptowährung wird in eine Tabelle namens `Inventory` geladen.
- Die Daten umfassen die folgenden Felder: `id`, `purchase_date` und `wph` (Watt pro Stunde).

Der Benutzer möchte eine Tabelle, die nach `id` die Gesamtkosten angibt, die für jede Schürfrunde bisher im Monat für den Stromverbrauch angefallen sind.

Dieser Wert soll aktualisiert werden, sooft das Diagrammobjekt aktualisiert wird. Die aktuellen Stromkosten belaufen sich auf $0,0678 pro kWh.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Inventory:
Load
*
Inline
[
id,purchase_date,wph
8188,1/7/2022,1123
8189,1/19/2022,1432
8190,2/28/2022,1227
8191,2/5/2022,1322
8192,3/16/2022,1273
8193,4/1/2022,1123
8194,5/7/2022,1342
8195,5/16/2022,2342
8196,6/15/2022,1231
8197,6/26/2022,1231
8198,7/9/2022,1123
8199,7/22/2022,1212
8200,7/23/2022,1223
8201,7/27/2022,1232
8202,8/2/2022,1232
8203,8/8/2022,1211
8204,8/19/2022,1243
8205,9/26/2022,1322
8206,10/14/2022,1133
8207,10/29/2022,1231
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu: `id`.

Erstellen Sie die folgende Kennzahl:

```
=(now(1)-monthstart(now(1)))*24*wph/1000*0.0678
```

Wenn das Diagrammobjekt am 6/22/2022 um 10:39:05 AM aktualisiert wurde, gibt es die folgenden Ergebnisse zurück:

Ergebnistabelle

id	=(now(1)-monthstart(now(1)))*24*wph/1000*0.0678
8188	$39.18
8189	$49.97
8190	$42.81
8191	$46.13
8192	$44.42
8193	$39.18
8194	$46.83
8195	$81.72
8196	$42.95
8197	$42.95
8198	$39.18
8199	$42.29
8200	$42.67
8201	$42.99
8202	$42.99
8203	$42.25
8204	$43.37
8205	$46.13
8206	$39.53

Der Benutzer möchte, dass die Objektergebnisse jedes Mal aktualisiert werden, wenn das Objekt aktualisiert wird. Daher wird das Argument `timer_mode` für Instanzen der Funktion `now()` in der Formel angegeben. Der Zeitstempel für den Start des Monats, der mittels Verwendung der Funktion `now()` als Zeitstempelargument in der Funktion `monthstart()` identifiziert wird, wird von der aktuellen Uhrzeit abgezogen, die von der Funktion `now()` identifiziert wird. Dies ergibt die gesamte Zeit, die bisher im Monat verstrichen ist, in Tagen.

Dieser Wert wird mit 24 (der Anzahl Stunden eines Tages) und dann mit dem Wert im Feld `wph` multipliziert.

Um den Wert von Watt pro Stunde in Kilowatt pro Stunde zu konvertieren, wird das Ergebnis durch 1000 dividiert, bevor es schließlich mit dem angegebenen Preis pro kWh multipliziert wird.

quarterend

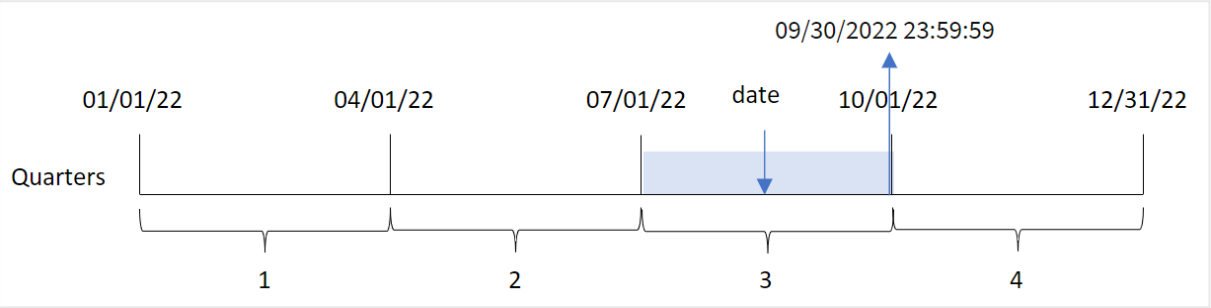
Diese Funktion liefert den Zeitstempel der letzten Millisekunde des Quartals, in dem **date** liegt. Das Ergebnis wird entsprechend dem im Skript definierten **DateFormat** formatiert.

Syntax:

```
QuarterEnd(date[, period_no[, first_month_of_year]])
```

Rückgabe Datentyp: dual

Diagramm der Funktion quarterend()



Die Funktion `quarterend()` bestimmt, in welches Quartal das Datum fällt. Sie gibt dann einen Zeitstempel im Datumsformat für die letzte Millisekunde des letzten Monats in diesem Quartal zurück. Der erste Monat des Jahres ist standardmäßig der Januar. Sie können aber ändern, welcher Monat als erster festgelegt wird, indem Sie das Argument `first_month_of_year` in der Funktion `quarterend()` verwenden.

> *Die Funktion `quarterend()` berücksichtigt die Systemvariable `FirstMonthOfYear` nicht. Das Jahr beginnt am 1. Januar, es sei denn, das Argument `first_month_of_year` wird verwendet, um dies zu ändern.*

Verwendung

Die Funktion `quarterend()` wird oft als Teil einer Formel verwendet, wenn in der Berechnung der Teil des Quartals verwendet werden soll, der noch nicht eingetreten ist. Beispiel: Sie möchten die gesamten, während des Quartals noch nicht fällig gewordenen Zinsen berechnen.

Argumente

Argument	Beschreibung
date	Datum oder Zeitstempel für die Evaluierung.
period_no	**period_no** ist eine ganze Zahl, wobei 0 für das Quartal steht, das **date** enthält. Negative Werte von **period_no** stehen für vorangehende, positive Werte für nachfolgende Quartale.
first_month_of_year	Wenn Sie mit abweichenden Geschäftsjahren arbeiten möchten, definieren Sie mit einer Zahl zwischen 2 und 12 für **first_month_of_year** einen anderen Startmonat.

Sie können die folgenden Werte verwenden, um den ersten Monat des Jahres im Argument `first_month_of_year` festzulegen.

Werte für first_month_of_year

Monat	Wert
Februar	2
März	3
April	4
Mai	5
Juni	6
Juli	7
August	8
September	9
Oktober	10
November	11
Dezember	12

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Funktionsbeispiele

Beispiel	Ergebnis
`quarterend('10/29/2005')`	Gibt 12/31/2005 23:59:59 zurück.
`quarterend('10/29/2005', -1)`	Gibt 09/30/2005 23:59:59 zurück.
`quarterend('10/29/2005', 0, 3)`	Gibt 11/30/2005 23:59:59 zurück.

Beispiel 1 – einfaches Beispiel

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz enthält eine Reihe von Transaktionen für 2022 und wird in eine Tabelle namens „`Transactions`“ geladen.
- Ein vorangehender load-Befehl, der Folgendes enthält:
    - Die Funktion `quarterend()`, die als das Feld „`end_of_quarter`“ festgelegt ist und einen Zeitstempel für das Ende des Quartals zurückgibt, in dem die Transaktionen stattfanden.
    - Die Funktion `timestamp()`, die als das Feld „`end_of_quarter_timestamp`“ festgelegt ist und den genauen Zeitstempel für das Ende des ausgewählten Quartals zurückgibt.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
       *,
       quarterend(date) as end_of_quarter,
       timestamp(quarterend(date)) as end_of_quarter_timestamp
       ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
```

```
8207,10/29/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `id`
- `date`
- `end_of_quarter`
- `end_of_quarter_timestamp`

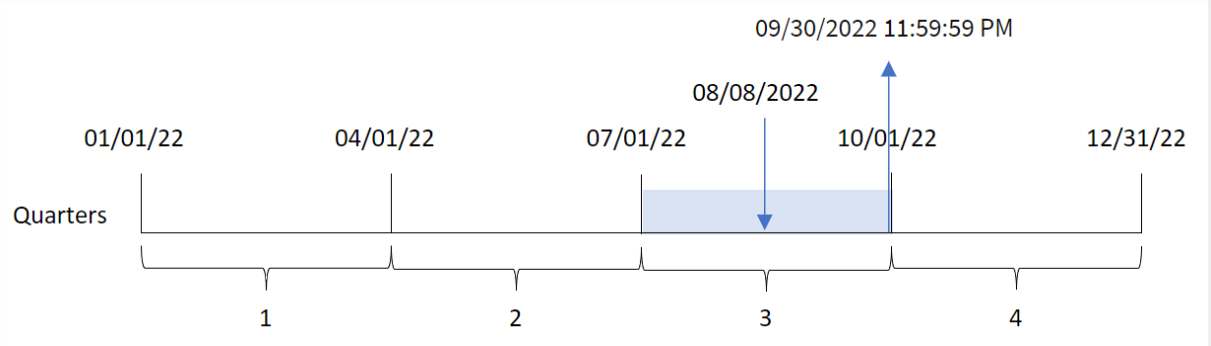
Ergebnistabelle

id	date	end_of_quarter	end_of_quarter_timestamp
8188	1/7/2022	03/31/2022	3/31/2022 11:59:59 PM
8189	1/19/2022	03/31/2022	3/31/2022 11:59:59 PM
8190	2/5/2022	03/31/2022	3/31/2022 11:59:59 PM
8191	2/28/2022	03/31/2022	3/31/2022 11:59:59 PM
8192	3/16/2022	03/31/2022	3/31/2022 11:59:59 PM
8193	4/1/2022	06/30/2022	6/30/2022 11:59:59 PM
8194	5/7/2022	06/30/2022	6/30/2022 11:59:59 PM
8195	5/16/2022	06/30/2022	6/30/2022 11:59:59 PM
8196	6/15/2022	06/30/2022	6/30/2022 11:59:59 PM
8197	6/26/2022	06/30/2022	6/30/2022 11:59:59 PM
8198	7/9/2022	09/30/2022	9/30/2022 11:59:59 PM
8199	7/22/2022	09/30/2022	9/30/2022 11:59:59 PM
8200	7/23/2022	09/30/2022	9/30/2022 11:59:59 PM
8201	7/27/2022	09/30/2022	9/30/2022 11:59:59 PM
8202	8/2/2022	09/30/2022	9/30/2022 11:59:59 PM
8203	8/8/2022	09/30/2022	9/30/2022 11:59:59 PM
8204	8/19/2022	09/30/2022	9/30/2022 11:59:59 PM
8205	9/26/2022	09/30/2022	9/30/2022 11:59:59 PM
8206	10/14/2022	12/31/2022	12/31/2022 11:59:59 PM
8207	10/29/2022	12/31/2022	12/31/2022 11:59:59 PM

Das Feld „`end_of_quarter`“ wird in der vorangehenden load-Anweisung erstellt, indem die Funktion `quarterend()` verwendet und das Datumsfeld als Argument der Funktion übergeben wird.

Die Funktion `quarterend()` identifiziert zuerst, in welches Quartal der Datumswert fällt, und gibt dann einen Zeitstempel für die letzte Millisekunde dieses Quartals zurück.

Diagramm der Funktion `quarterend()`, in dem das Quartalsende für Transaktion 8203 identifiziert ist



Transaktion 8203 fand am 8. August statt. Die Funktion `quarterend()` identifiziert, dass die Transaktion im dritten Quartal stattfand und gibt die letzte Millisekunde dieses Quartals zurück. Dies ist der 30. September um 11:59:59 PM.

Beispiel 2 – period_no

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Datensatz mit einer Reihe von Transaktionen für 2022, der in eine Tabelle namens „`Transactions`“ geladen wird
- Eine vorangehende load-Anweisung, die Folgendes enthält:
  - Die Funktion `quarterend()`, die als das Feld „`previous_quarter_end`“ festgelegt ist und einen Zeitstempel für das Ende des Quartals vor dem Quartal zurückgibt, in dem die Transaktion stattfand.
  - Die Funktion `timestamp()`, die als das Feld „`previous_end_of_quarter_timestamp`“ festgelegt ist und den genauen Zeitstempel für das Ende des Quartals vor dem Quartal zurückgibt, in dem die Transaktion stattfand.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
       *,
       quarterend(date, -1) as previous_quarter_end,
       timestamp(quarterend(date, -1)) as previous_quarter_end_timestamp
       ;
Load
```

```
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `id`
- `date`
- `previous_quarter_end`
- `previous_quarter_end_timestamp`

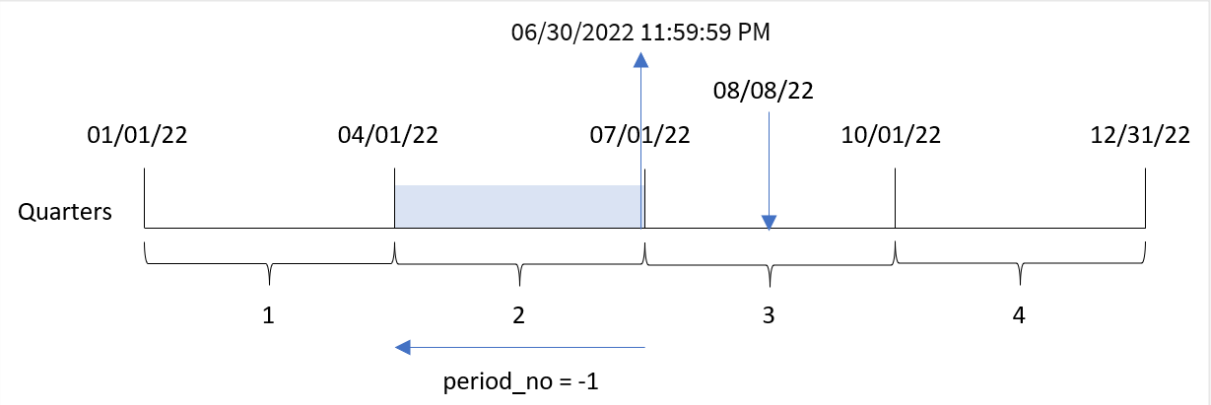
Ergebnistabelle

id	date	previous_quarter_end	previous_quarter_end_timestamp
8188	1/7/2022	12/31/2021	12/31/2021 11:59:59 PM
8189	1/19/2022	12/31/2021	12/31/2021 11:59:59 PM
8190	2/5/2022	12/31/2021	12/31/2021 11:59:59 PM
8191	2/28/2022	12/31/2021	12/31/2021 11:59:59 PM
8192	3/16/2022	12/31/2021	12/31/2021 11:59:59 PM
8193	4/1/2022	03/31/2022	3/31/2022 11:59:59 PM
8194	5/7/2022	03/31/2022	3/31/2022 11:59:59 PM

id	date	previous_quarter_end	previous_quarter_end_timestamp
8195	5/16/2022	03/31/2022	3/31/2022 11:59:59 PM
8196	6/15/2022	03/31/2022	3/31/2022 11:59:59 PM
8197	6/26/2022	03/31/2022	3/31/2022 11:59:59 PM
8198	7/9/2022	06/30/2022	6/30/2022 11:59:59 PM
8199	7/22/2022	06/30/2022	6/30/2022 11:59:59 PM
8200	7/23/2022	06/30/2022	6/30/2022 11:59:59 PM
8201	7/27/2022	06/30/2022	6/30/2022 11:59:59 PM
8202	8/2/2022	06/30/2022	6/30/2022 11:59:59 PM
8203	8/8/2022	06/30/2022	6/30/2022 11:59:59 PM
8204	8/19/2022	06/30/2022	6/30/2022 11:59:59 PM
8205	9/26/2022	06/30/2022	6/30/2022 11:59:59 PM
8206	10/14/2022	09/30/2022	9/30/2022 11:59:59 PM
8207	10/29/2022	09/30/2022	9/30/2022 11:59:59 PM

Da eine `period_no` von `-1` als Versatzargument in der Funktion `quarterend()` verwendet wird, identifiziert die Funktion zuerst das Quartal, in dem die Transaktionen stattfinden. Dann geht sie ein Quartal zurück und identifiziert die letzte Millisekunde dieses Quartals.

Diagramm der Funktion `quarterend()` mit einer „period_no“ von -1.



Transaktion 8203 fand am 8. August statt. Die Funktion `quarterend()` identifiziert, dass das Quartal vor dem Quartal, in dem die Transaktion stattfand, zwischen dem 1. April und 30. Juni lag. Die Funktion gibt dann die letzte Millisekunde für dieses Quartal zurück, den 30. Juni um 11:59:59 PM.

Beispiel 3 – first_month_of_year

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Datensatz mit einer Reihe von Transaktionen für 2022, der in eine Tabelle namens „`Transactions`“ geladen wird
- Eine vorangehende load-Anweisung, die Folgendes enthält:
    - Die Funktion `quarterend()`, die als das Feld „`end_of_quarter`“ festgelegt ist und einen Zeitstempel für das Ende des Quartals zurückgibt, in dem die Transaktionen stattfanden.
    - Die Funktion `timestamp()`, die als das Feld „`end_of_quarter_timestamp`“ festgelegt ist und den genauen Zeitstempel für das Ende des ausgewählten Quartals zurückgibt.

In diesem Beispiel legt die Unternehmensrichtlinie jedoch fest, dass das Geschäftsjahr am 1. März beginnt.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
       *,
       quarterend(date, 0, 3) as end_of_quarter,
       timestamp(quarterend(date, 0, 3)) as end_of_quarter_timestamp
       ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
```

```
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

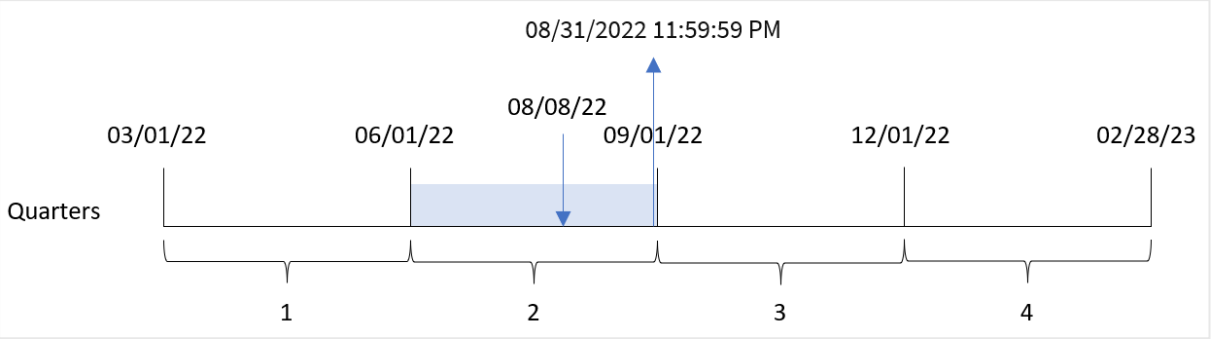
Ergebnisse

Ergebnistabelle

id	date	end_of_quarter	end_of_quarter_timestamp
8188	1/7/2022	02/28/2022	2/28/2022 11:59:59 PM
8189	1/19/2022	02/28/2022	2/28/2022 11:59:59 PM
8190	2/5/2022	02/28/2022	2/28/2022 11:59:59 PM
8191	2/28/2022	02/28/2022	2/28/2022 11:59:59 PM
8192	3/16/2022	05/31/2022	5/31/2022 11:59:59 PM
8193	4/1/2022	05/31/2022	5/31/2022 11:59:59 PM
8194	5/7/2022	05/31/2022	5/31/2022 11:59:59 PM
8195	5/16/2022	05/31/2022	5/31/2022 11:59:59 PM
8196	6/15/2022	08/31/2022	8/31/2022 11:59:59 PM
8197	6/26/2022	08/31/2022	8/31/2022 11:59:59 PM
8198	7/9/2022	08/31/2022	8/31/2022 11:59:59 PM
8199	7/22/2022	08/31/2022	8/31/2022 11:59:59 PM
8200	7/23/2022	08/31/2022	8/31/2022 11:59:59 PM
8201	7/27/2022	08/31/2022	8/31/2022 11:59:59 PM
8202	8/2/2022	08/31/2022	8/31/2022 11:59:59 PM
8203	8/8/2022	08/31/2022	8/31/2022 11:59:59 PM
8204	8/19/2022	08/31/2022	8/31/2022 11:59:59 PM
8205	9/26/2022	11/30/2022	11/30/2022 11:59:59 PM
8206	10/14/2022	11/30/2022	11/30/2022 11:59:59 PM
8207	10/29/2022	11/30/2022	11/30/2022 11:59:59 PM

Da das Argument `first_month_of_year` von `3` in der Funktion `quarterend()` verwendet wird, wird der Start des Jahres vom 1. Januar auf den 1. März verschoben.

Diagramm der Funktion `quarterend()` mit März als erstem Monat des Jahres



Transaktion 8203 fand am 8. August statt. Da der Anfang des Jahres der 1. März ist, sind die Quartale des Jahres Mär-Mai, Jun-Aug, Sep-Nov und Dez-Feb.

Die Funktion `quarterend()` identifiziert, dass die Transaktion im Quartal zwischen Anfang Juni und Ende August stattfand und gibt die letzte Millisekunde dieses Quartals zurück. Dies ist der 31. August um 11:59:59 PM.

Beispiel 4 – Diagrammobjektbeispiel

Ladeskript und Diagrammformel

Übersicht

Es werden derselbe Datensatz und dasselbe Szenario wie im ersten Beispiel verwendet.

In diesem Beispiel wird jedoch der unveränderte Datensatz in die Anwendung geladen. Die Berechnung, die einen Zeitstempel für das Ende des Quartals zurückgibt, in dem die Transaktionen stattfanden, wird als Kennzahl in einem Diagramm in der App erstellt.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
```

```
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `id`
- `date`

Um das Enddatum des Quartals zu berechnen, in dem eine Transaktion stattfindet, erstellen Sie die folgenden Kennzahlen:

- `=quarterend(date)`
- `=timestamp(quarterend(date))`

Ergebnistabelle

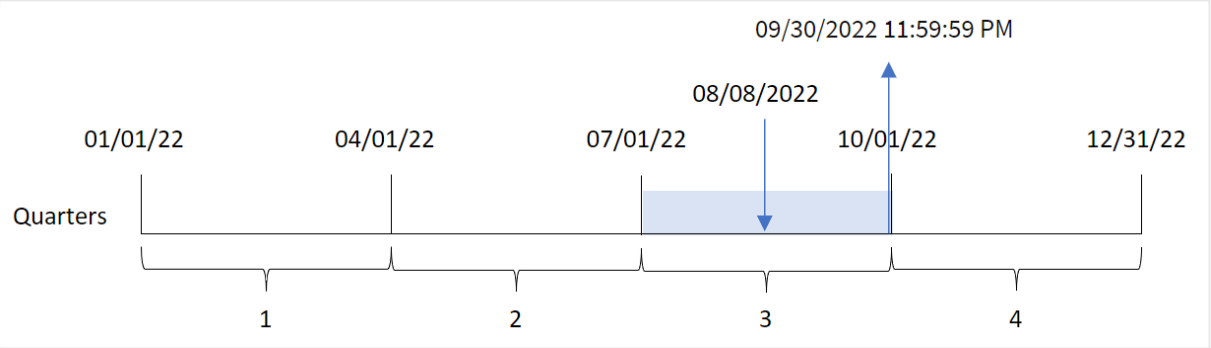
id	date	=quarterend(date)	=timestamp(quarterend(date))
8188	1/7/2022	03/31/2022	3/31/2022 11:59:59 PM
8189	1/19/2022	03/31/2022	3/31/2022 11:59:59 PM
8190	2/5/2022	03/31/2022	3/31/2022 11:59:59 PM
8191	2/28/2022	03/31/2022	3/31/2022 11:59:59 PM
8192	3/16/2022	03/31/2022	3/31/2022 11:59:59 PM
8193	4/1/2022	06/30/2022	6/30/2022 11:59:59 PM
8194	5/7/2022	06/30/2022	6/30/2022 11:59:59 PM
8195	5/16/2022	06/30/2022	6/30/2022 11:59:59 PM
8196	6/15/2022	06/30/2022	6/30/2022 11:59:59 PM
8197	6/26/2022	06/30/2022	6/30/2022 11:59:59 PM
8198	7/9/2022	09/30/2022	9/30/2022 11:59:59 PM
8199	7/22/2022	09/30/2022	9/30/2022 11:59:59 PM
8200	7/23/2022	09/30/2022	9/30/2022 11:59:59 PM

id	date	=quarterend(date)	=timestamp(quarterend(date))
8201	7/27/2022	09/30/2022	9/30/2022 11:59:59 PM
8202	8/2/2022	09/30/2022	9/30/2022 11:59:59 PM
8203	8/8/2022	09/30/2022	9/30/2022 11:59:59 PM
8204	8/19/2022	09/30/2022	9/30/2022 11:59:59 PM
8205	9/26/2022	09/30/2022	9/30/2022 11:59:59 PM
8206	10/14/2022	12/31/2022	12/31/2022 11:59:59 PM
8207	10/29/2022	12/31/2022	12/31/2022 11:59:59 PM

Das Feld „`end_of_quarter`“ wird in der vorangehenden load-Anweisung erstellt, indem die Funktion `quarterend()` verwendet und das Datumsfeld als Argument der Funktion übergeben wird.

Die Funktion `quarterend()` identifiziert zuerst, in welches Quartal der Datumswert fällt, und gibt dann einen Zeitstempel für die letzte Millisekunde dieses Quartals zurück.

Diagramm der Funktion `quarterend()`, in dem das Quartalsende für Transaktion 8203 identifiziert ist



Transaktion 8203 fand am 8. August statt. Die Funktion `quarterend()` identifiziert, dass die Transaktion im dritten Quartal stattfand und gibt die letzte Millisekunde dieses Quartals zurück. Dies ist der 30. September um 11:59:59 PM.

Beispiel 5 – Szenario

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz wird in eine Tabelle namens „`Employee_Expenses`“ geladen. Die Tabelle enthält die folgenden Felder:

- Mitarbeiter-IDs
- Mitarbeiternamen
- Die durchschnittlichen täglichen Spesenanträge pro Mitarbeiter.

Der Endbenutzer möchte ein Diagrammobjekt, das nach Mitarbeiter-ID und Mitarbeitername die geschätzten Spesenanträge anzeigt, die für das restliche Quartal noch anfallen werden. Das Geschäftsjahr beginnt im Januar.

Ladeskript

```
Employee_Expenses:
Load
*
Inline
[
employee_id,employee_name,avg_daily_claim
182,Mark, $15
183,Deryck, $12.5
184,Dexter, $12.5
185,Sydney,$27
186,Agatha,$18
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `employee_id`
- `employee_name`

Erstellen Sie die folgende Kennzahl, um die kumulierten Zinsen zu berechnen:

- `=(quarterend(today(1))-today(1))*avg_daily_claim`

Legen Sie das **Zahlenformat** der Kennzahl auf **Währung** fest.

Ergebnistabelle

employee_id	employee_name	=(quarterend(today(1))-today(1))*avg_daily_claim
182	Mark	$480.00
183	Deryck	$400.00
184	Dexter	$400.00
185	Sydney	$864.00
186	Agatha	$576.00

Die Funktion `quarterend()` verwendet das aktuelle Datum als einziges Argument und gibt das Enddatum des aktuellen Monats zurück. Dann zieht sie das aktuelle Datum vom Jahresenddatum ab, und die Formel gibt die Anzahl der im Monat verbleibenden Tage zurück.

Dieser Wert wird dann mit den durchschnittlichen täglichen Spesenanträgen der einzelnen Mitarbeitern multipliziert, um den geschätzten Spesenbetrag pro Mitarbeiter für das verbleibende Quartal zu berechnen.

quartername

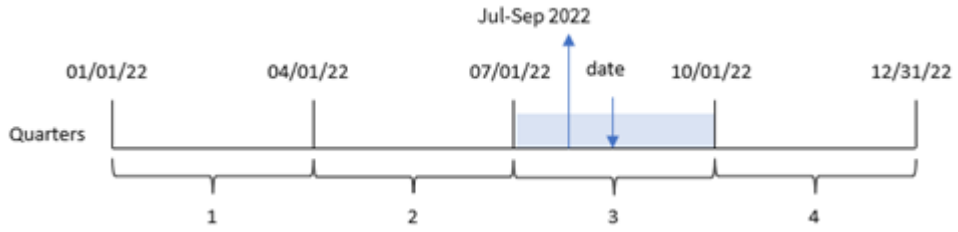
Diese Funktion liefert den Zeitstempel der ersten Millisekunde des ersten Tags des Quartals. Das Ergebnis wird als Kombination von Monaten (entsprechend der Skriptvariablen **MonthNames**) und Jahr formatiert.

Syntax:

```
QuarterName(date[, period_no[, first_month_of_year]])
```

Rückgabe Datentyp: dual

Diagramm der Funktion `quartername()`



Die Funktion `quartername()` bestimmt, in welches Quartal das Datum fällt. Dann gibt sie einen Wert zurück, der die Start- und Endmonate dieses Quartals sowie das Jahr zeigt. Der zugrunde liegende numerische Wert dieses Ergebnisses ist die erste Millisekunde des Quartals.

Argumente

Argument	Beschreibung
date	Datum oder Zeitstempel für die Evaluierung.
period_no	**period_no** ist eine ganze Zahl, wobei 0 für das Quartal steht, das **date** enthält. Negative Werte von **period_no** stehen für vorangehende, positive Werte für nachfolgende Quartale.
first_month_of_year	Wenn Sie mit abweichenden Geschäftsjahren arbeiten möchten, definieren Sie mit einer Zahl zwischen 2 und 12 für **first_month_of_year** einen anderen Startmonat.

Verwendung

Die Funktion `quartername()` ist nützlich, wenn Sie Aggregierungen nach Quartal vergleichen möchten. Das ist beispielsweise der Fall, wenn Sie den Gesamtumsatz von Produkten nach Quartal anzeigen möchten.

Diese Funktion kann im Ladeskript verwendet werden, um ein Feld in einer Master-Kalender-Tabelle zu erstellen. Alternativ kann sie direkt als berechnete Dimension in einem Diagramm verwendet werden.

In diesen Beispielen wird das Datumsformat MM/TT/JJJJ verwendet. Das Datumsformat wird im Befehl `SET DateFormat` oben in Ihrem Datenladeskript angegeben. Ändern Sie das Format in den Beispielen nach Bedarf.

Funktionsbeispiele

Beispiel	Ergebnis
`quartername('10/29/2013')`	Gibt Oct-Dec 2013 zurück.
`quartername('10/29/2013', -1)`	Gibt Jul-Sep 2013 zurück.
`quartername('10/29/2013', 0, 3)`	Gibt Sep-Nov 2013 zurück.

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Beispiel 1 – Datum ohne zusätzliche Argumente

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz enthält eine Reihe von Transaktionen für 2022 und wird in eine Tabelle namens `Transactions` geladen.
- Das Datumsfeld wird im Format der Systemvariablen `DateFormat` (MM/TT/JJJJ) bereitgestellt.
- Es wird ein Feld `transaction_quarter` erstellt, das das Quartal zurückgibt, in dem die Transaktionen stattfanden.

Fügen Sie bei Bedarf hier weiteren Text mit Listen usw. hinzu.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';
SET MonthNames='Jan;Feb;Mar;Apr;May;Jun;Jul;Aug;Sep;Oct;Nov;Dec';
```

```
Transactions:
    Load
        *,
        quartername(date) as transaction_quarter
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `transaction_quarter`

Ergebnistabelle

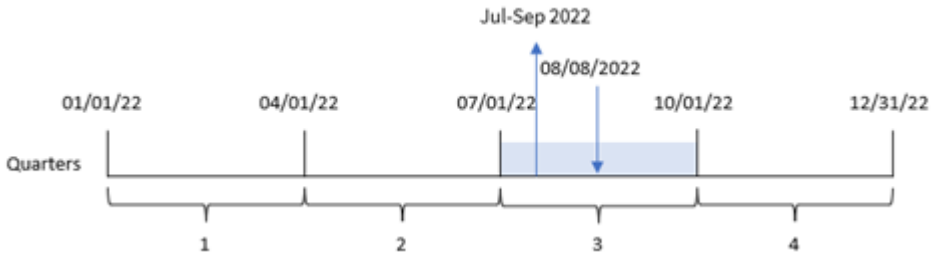
date	transaction_quarter
1/7/2022	Jan-Mar 2022
1/19/2022	Jan-Mar 2022
2/5/2022	Jan-Mar 2022
2/28/2022	Jan-Mar 2022
3/16/2022	Jan-Mar 2022

date	transaction_quarter
4/1/2022	Apr-Jun 2022
5/7/2022	Apr-Jun 2022
5/16/2022	Apr-Jun 2022
6/15/2022	Apr-Jun 2022
6/26/2022	Apr-Jun 2022
7/9/2022	Jul-Sep 2022
7/22/2022	Jul-Sep 2022
7/23/2022	Jul-Sep 2022
7/27/2022	Jul-Sep 2022
8/2/2022	Jul-Sep 2022
8/8/2022	Jul-Sep 2022
8/19/2022	Jul-Sep 2022
9/26/2022	Jul-Sep 2022
10/14/2022	Oct-Dec 2022
10/29/2022	Oct-Dec 2022

Das Feld `transaction_quarter` wird im vorangehenden load-Befehl erstellt, indem die Funktion `quartername ()` verwendet und das Datumsfeld als Argument der Funktion übergeben wird.

Die Funktion `quartername()` identifiziert zunächst das Quartal, in das der Datumswert fällt. Dann gibt sie einen Wert zurück, der die Start- und Endmonate dieses Quartals sowie das Jahr zeigt.

Diagramm der Funktion `quartername()`, Beispiel ohne zusätzliche Argumente



Transaktion 8203 fand am 8. August 2022 statt. Die Funktion `quartername()` identifiziert, dass die Transaktion im dritten Quartal stattfand, und gibt daher Jul-Sep 2022 zurück. Die Monate werden im Format der Systemvariablen `MonthNames` angezeigt.

Beispiel 2 – Datum mit Argument „period_no“

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Derselbe Datensatz und dasselbe Szenario wie im ersten Beispiel.
- Es wird ein Feld `previous_quarter` erstellt, das das Quartal vor dem Quartal zurückgibt, in dem die Transaktionen stattfanden.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';
SET MonthNames='Jan;Feb;Mar;Apr;May;Jun;Jul;Aug;Sep;Oct;Nov;Dec';

Transactions:
    Load
        *,
        quartername(date,-1) as previous_quarter
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

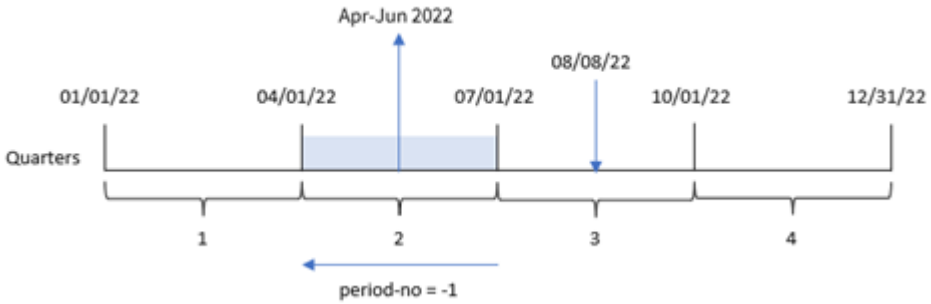
- `date`
- `previous_quarter`

Ergebnistabelle

date	previous_quarter
1/7/2022	Oct-Dec 2021
1/19/2022	Oct-Dec 2021
2/5/2022	Oct-Dec 2021
2/28/2022	Oct-Dec 2021
3/16/2022	Oct-Dec 2021
4/1/2022	Jan-Mar 2022
5/7/2022	Jan-Mar 2022
5/16/2022	Jan-Mar 2022
6/15/2022	Jan-Mar 2022
6/26/2022	Jan-Mar 2022
7/9/2022	Apr-Jun 2022
7/22/2022	Apr-Jun 2022
7/23/2022	Apr-Jun 2022
7/27/2022	Apr-Jun 2022
8/2/2022	Apr-Jun 2022
8/8/2022	Apr-Jun 2022
8/19/2022	Apr-Jun 2022
9/26/2022	Apr-Jun 2022
10/14/2022	Jul-Sep 2022
10/29/2022	Jul-Sep 2022

Da in diesem Fall eine `period_no` von -1 als Versatzargument in der Funktion `quartername()` verwendet wurde, identifiziert die Funktion zuerst, dass die Transaktionen im dritten Quartal stattfanden. Dann geht sie ein Quartal zurück und gibt einen Wert zurück, der die Start- und Endmonate dieses Quartals sowie das Jahr zeigt.

Diagramm der Funktion `quartername()`, Beispiel „period_no“



Transaktion 8203 fand am 8. August statt. Die Funktion `quartername()` identifiziert, dass das Quartal vor dem Quartal, in dem die Transaktion stattfand, zwischen dem 1. April und 30. Juni lag. Somit wird Apr-Jun 2022 zurückgegeben.

Beispiel 3 – Datum mit dem Argument „first_week_day“

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript verwendet den gleichen Datensatz und das gleiche Szenario wie das erste Beispiel. In diesem Beispiel müssen wir aber den 1. März als Beginn des Geschäftsjahres festlegen.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';
SET MonthNames='Jan;Feb;Mar;Apr;May;Jun;Jul;Aug;Sep;Oct;Nov;Dec';

Transactions:
    Load
        *,
        quartername(date,0,3) as transaction_quarter
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
```

```
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `transaction_quarter`

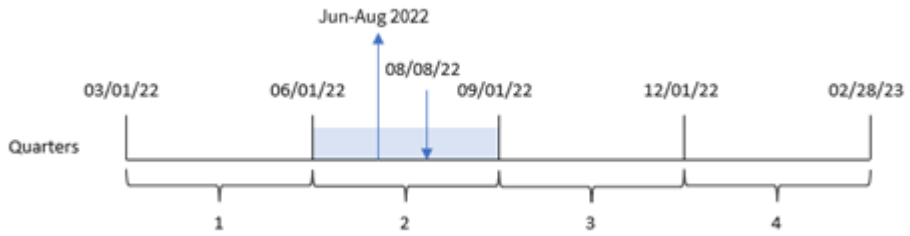
Ergebnistabelle

date	**transaction_quarter**
1/7/2022	Dec-Feb 2021
1/19/2022	Dec-Feb 2021
2/5/2022	Dec-Feb 2021
2/28/2022	Dec-Feb 2021
3/16/2022	Mar-May 2022
4/1/2022	Mar-May 2022
5/7/2022	Mar-May 2022
5/16/2022	Mar-May 2022
6/15/2022	Jun-Aug 2022
6/26/2022	Jun-Aug 2022
7/9/2022	Jun-Aug 2022
7/22/2022	Jun-Aug 2022
7/23/2022	Jun-Aug 2022
7/27/2022	Jun-Aug 2022
8/2/2022	Jun-Aug 2022
8/8/2022	Jun-Aug 2022
8/19/2022	Jun-Aug 2022

date	transaction_quarter
9/26/2022	Sep-Nov 2022
10/14/2022	Sep-Nov 2022
10/29/2022	Sep-Nov 2022

Da in diesem Fall das Argument `first_month_of_year` von 3 in der Funktion `quartername()` verwendet wird, wird der Start des Jahres vom 1. Januar zum 1. März verschoben. Somit werden die Quartale des Jahres in März-Mai, Juni-August, September-November und Dezember-Februar unterteilt.

Diagramm der Funktion `quartername()`, Beispiel „first_week_day“



Transaktion 8203 fand am 8. August statt. Die Funktion `quartername()` identifiziert, dass die Transaktion im zweiten Quartal stattfand, zwischen Anfang Juni und Ende August. Somit wird Jun-Aug 2022 zurückgegeben.

Beispiel 4 – Diagrammobjektbeispiel

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript verwendet den gleichen Datensatz und das gleiche Szenario wie das erste Beispiel.

In diesem Beispiel wird jedoch der unveränderte Datensatz in die Anwendung geladen. Die Berechnung, die einen Zeitstempel für das Ende des Quartals zurückgibt, in dem die Transaktionen stattfanden, wird als Kennzahl in einem Diagrammobjekt der Anwendung erstellt.

Ladeskript

```
Transactions:
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
```

```
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu: `date`.

Erstellen Sie die folgende Kennzahl:

```
=quartername(date)
```

Ergebnistabelle

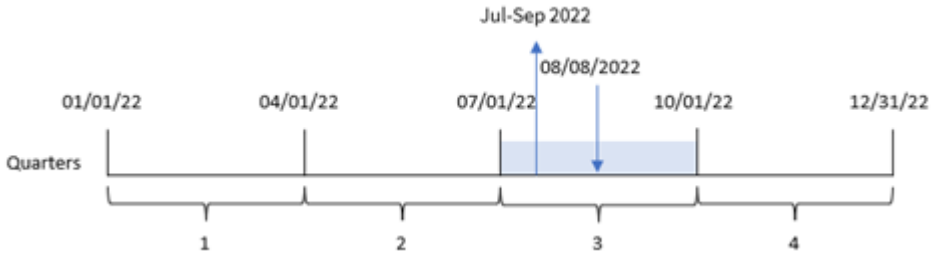
date	=quartername(date)
1/7/2022	Jan-Mar 2022
1/19/2022	Jan-Mar 2022
2/5/2022	Jan-Mar 2022
2/28/2022	Jan-Mar 2022
3/16/2022	Jan-Mar 2022
4/1/2022	Apr-Jun 2022
5/7/2022	Apr-Jun 2022
5/16/2022	Apr-Jun 2022
6/15/2022	Apr-Jun 2022
6/26/2022	Apr-Jun 2022
7/9/2022	Jul-Sep 2022
7/22/2022	Jul-Sep 2022
7/23/2022	Jul-Sep 2022
7/27/2022	Jul-Sep 2022

date	=quartername(date)
8/2/2022	Jul-Sep 2022
8/8/2022	Jul-Sep 2022
8/19/2022	Jul-Sep 2022
9/26/2022	Jul-Sep 2022
10/14/2022	Oct-Dec 2022
10/29/2022	Oct-Dec 2022

Die Kennzahl `transaction_quarter` wird im Diagrammobjekt erstellt, indem die Funktion `quartername()` verwendet und das Feld `date` als Argument der Funktion übergeben wird.

Die Funktion `quartername()` identifiziert zunächst das Quartal, in das der Datumswert fällt. Dann gibt sie einen Wert zurück, der die Start- und Endmonate dieses Quartals sowie das Jahr zeigt.

Diagramm der Funktion `quartername()`, Diagrammobjektbeispiel



Transaktion 8203 fand am 8. August 2022 statt. Die Funktion `quartername()` identifiziert, dass die Transaktion im dritten Quartal stattfand, und gibt daher Jul-Sep 2022 zurück. Die Monate werden im Format der Systemvariablen `MonthNames` angezeigt.

Beispiel 5 – Szenario

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz enthält eine Reihe von Transaktionen für 2022 und wird in eine Tabelle namens `Transactions` geladen.
- Das Datumsfeld wird im Format der Systemvariablen `DateFormat` (MM/TT/JJJJ) bereitgestellt.

Der Endbenutzer möchte ein Diagrammobjekt, das den Gesamtumsatz nach Quartal für die Transaktionen darstellt. Dies ist selbst dann möglich, wenn diese Dimension nicht im Datenmodell verfügbar ist. In diesem Fall wird die Funktion `quartername()` als berechnete Dimension im Diagramm verwendet.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8188,'1/7/2022',17.17
8189,'1/19/2022',37.23
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'5/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'6/26/2022',45.89
8198,'7/9/2022',36.23
8199,'7/22/2022',25.66
8200,'7/23/2022',82.77
8201,'7/27/2022',69.98
8202,'8/2/2022',76.11
8203,'8/8/2022',25.12
8204,'8/19/2022',46.23
8205,'9/26/2022',84.21
8206,'10/14/2022',96.24
8207,'10/29/2022',67.67
];
```

Ergebnisse

Gehen Sie folgendermaßen vor:

1. Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle.
2. Erstellen Sie eine berechnete Dimension anhand der folgenden Formel:
   `=quartername(date)`
3. Berechnen Sie dann den Gesamtumsatz anhand der folgenden Aggregierungskennzahl:
   `=sum(amount)`
4. Legen Sie das **Zahlenformat** der Kennzahl auf **Währung** fest.

Ergebnistabelle

=quartername(date)	=sum(amount)
Jul-Sep 2022	$446.31
Apr-Jun 2022	$351.48
Jan-Mar 2022	$253.89
Oct-Dec 2022	$163.91

quarterstart

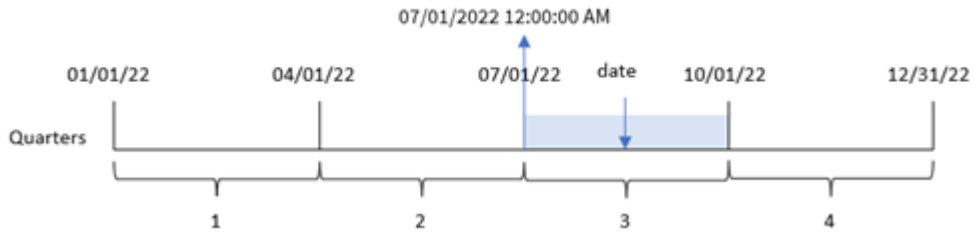
Diese Funktion liefert den Zeitstempel der ersten Millisekunde des Quartals, in dem **date** liegt. Das Ergebnis wird entsprechend dem im Skript definierten **DateFormat** formatiert.

Syntax:

```
QuarterStart(date[, period_no[, first_month_of_year]])
```

Rückgabe Datentyp: dual

Diagramm der Funktion quarterstart()



Die Funktion `quarterstart()` bestimmt, in welches Quartal das `date` fällt. Sie gibt dann einen Zeitstempel im Datumsformat für die erste Millisekunde im ersten Monat dieses Quartals zurück.

Argumente

Argument	Beschreibung
date	Datum oder Zeitstempel für die Evaluierung.
period_no	**period_no** ist eine ganze Zahl, wobei 0 für das Quartal steht, das **date** enthält. Negative Werte von **period_no** stehen für vorangehende, positive Werte für nachfolgende Quartale.
first_month_of_year	Wenn Sie mit abweichenden Geschäftsjahren arbeiten möchten, definieren Sie mit einer Zahl zwischen 2 und 12 für **first_month_of_year** einen anderen Startmonat.

Verwendung

Die Funktion `quarterstart()` wird in der Regel als Teil einer Formel verwendet, wenn in der Berechnung der Teil des Quartals verwendet werden soll, der bereits verstrichen ist. Zum Beispiel kann sie von einem Benutzer verwendet werden, um die Zinsen zu berechnen, die in einem Quartal bis dato aufgelaufen sind.

Funktionsbeispiele

Beispiel	Ergebnis
`quarterstart('10/29/2005')`	Gibt 10/01/2005 zurück.
`quarterstart('10/29/2005', -1 )`	Gibt 07/01/2005 zurück.
`quarterstart('10/29/2005', 0, 3)`	Gibt 09/01/2005 zurück.

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Beispiel 1 – keine zusätzlichen Argumente

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz enthält eine Reihe von Transaktionen für 2022 und wird in eine Tabelle namens `Transactions` geladen.
- Das Datumsfeld wird im Format der Systemvariablen `DateFormat` (MM/TT/JJJJ) bereitgestellt.
- Es wird ein Feld `start_of_quarter` erstellt, das einen Zeitstempel für den Start des Quartals zurückgibt, in dem die Transaktionen stattfanden.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        quarterstart(date) as start_of_quarter,
        timestamp(quarterstart(date)) as start_of_quarter_timestamp
        ;
```

```
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `start_of_quarter`
- `start_of_quarter_timestamp`

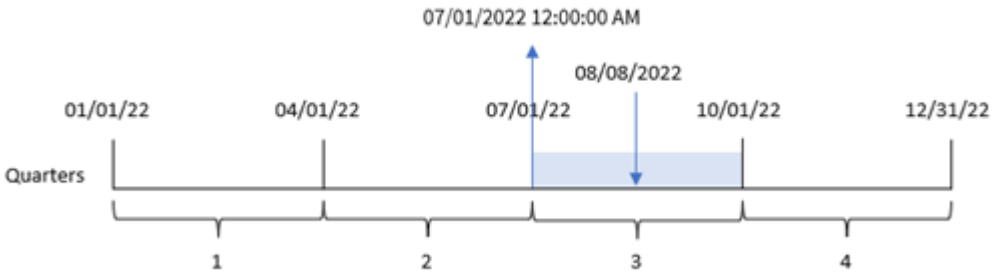
Ergebnistabelle

date	start_of_quarter	start_of_quarter_timestamp
1/7/2022	01/01/2022	1/1/2022 12:00:00 AM
1/19/2022	01/01/2022	1/1/2022 12:00:00 AM
2/5/2022	01/01/2022	1/1/2022 12:00:00 AM
2/28/2022	01/01/2022	1/1/2022 12:00:00 AM
3/16/2022	01/01/2022	1/1/2022 12:00:00 AM
4/1/2022	04/01/2022	4/1/2021 12:00:00 AM
5/7/2022	04/01/2022	4/1/2021 12:00:00 AM
5/16/2022	04/01/2022	4/1/2021 12:00:00 AM

date	start_of_quarter	start_of_quarter_timestamp
6/15/2022	04/01/2022	4/1/2021 12:00:00 AM
6/26/2022	04/01/2022	4/1/2021 12:00:00 AM
7/9/2022	07/01/2022	7/1/2021 12:00:00 AM
7/22/2022	07/01/2022	7/1/2021 12:00:00 AM
7/23/2022	07/01/2022	7/1/2021 12:00:00 AM
7/27/2022	07/01/2022	7/1/2021 12:00:00 AM
8/2/2022	07/01/2022	7/1/2021 12:00:00 AM
8/8/2022	07/01/2022	7/1/2021 12:00:00 AM
8/19/2022	07/01/2022	7/1/2021 12:00:00 AM
9/26/2022	07/01/2022	7/1/2021 12:00:00 AM
10/14/2022	10/01/2022	10/1/2022 12:00:00 AM
10/29/2022	10/01/2022	10/1/2022 12:00:00 AM

Das Feld `start_of_quarter` wird im vorangehenden load-Befehl erstellt, indem die Funktion `quarterstart()` verwendet und das Datumsfeld als Argument der Funktion übergeben wird. Die Funktion `quarterstart()` identifiziert anfänglich das Quartal, in das der Datumswert fällt. Sie gibt dann einen Zeitstempel für die erste Millisekunde dieses Quartals zurück.

Diagramm der Funktion `quarterstart()`, Beispiel ohne zusätzliche Argumente



Transaktion 8203 fand am 8. August statt. Die Funktion `quarterstart()` identifiziert, dass die Transaktion im dritten Quartal stattfand und gibt die erste Millisekunde dieses Quartals zurück. Dies ist der 1. Juli um 12:00:00 AM.

Beispiel 2 – period_no

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Derselbe Datensatz und dasselbe Szenario wie im ersten Beispiel.
- Es wird ein Feld `previous_quarter_start` erstellt, das den Zeitstempel für den Start des Quartals vor dem Quartal zurückgibt, in dem die Transaktion stattfand.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        quarterstart(date,-1) as previous_quarter_start,
        timestamp(quarterstart(date,-1)) as previous_quarter_start_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

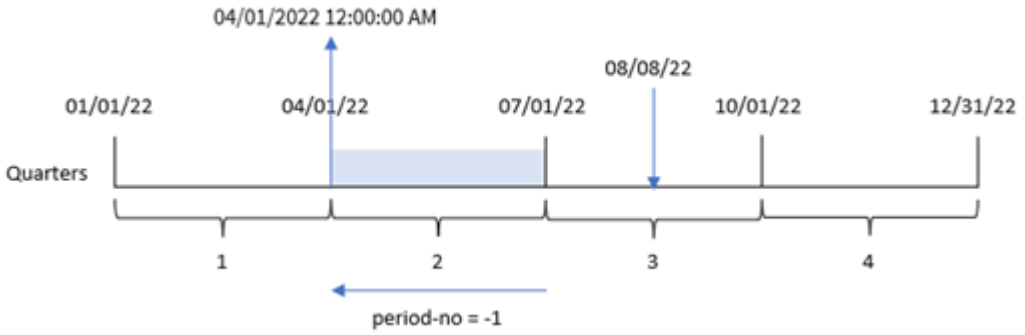
- `date`
- `previous_quarter_start`
- `previous_quarter_start_timestamp`

Ergebnistabelle

date	previous_quarter_start	previous_quarter_start_timestamp
1/7/2022	10/01/2021	10/1/2021 12:00:00 AM
1/19/2022	10/01/2021	10/1/2021 12:00:00 AM
2/5/2022	10/01/2021	10/1/2021 12:00:00 AM
2/28/2022	10/01/2021	10/1/2021 12:00:00 AM
3/16/2022	10/01/2021	10/1/2021 12:00:00 AM
4/1/2022	01/01/2022	1/1/2022 12:00:00 AM
5/7/2022	01/01/2022	1/1/2022 12:00:00 AM
5/16/2022	01/01/2022	1/1/2022 12:00:00 AM
6/15/2022	01/01/2022	1/1/2022 12:00:00 AM
6/26/2022	01/01/2022	1/1/2022 12:00:00 AM
7/9/2022	04/01/2022	4/1/2021 12:00:00 AM
7/22/2022	04/01/2022	4/1/2021 12:00:00 AM
7/23/2022	04/01/2022	4/1/2021 12:00:00 AM
7/27/2022	04/01/2022	4/1/2021 12:00:00 AM
8/2/2022	04/01/2022	4/1/2021 12:00:00 AM
8/8/2022	04/01/2022	4/1/2021 12:00:00 AM
8/19/2022	04/01/2022	4/1/2021 12:00:00 AM
9/26/2022	04/01/2022	4/1/2021 12:00:00 AM
10/14/2022	07/01/2022	7/1/2022 12:00:00 AM
10/29/2022	07/01/2022	7/1/2022 12:00:00 AM

Da in diesem Fall eine `period_no` von -1 als Versatzargument in der Funktion `quarterstart()` verwendet wurde, identifiziert die Funktion zuerst das Quartal, in dem die Transaktionen stattfanden. Dann geht sie ein Quartal zurück und identifiziert die erste Millisekunde dieses Quartals.

Diagramm der Funktion `quarterstart()`, Beispiel „period_no“



Transaktion 8203 fand am 8. August statt. Die Funktion `quarterstart()` identifiziert, dass das Quartal vor dem Quartal, in dem die Transaktion stattfand, zwischen dem 1. April und 30. Juni lag. Dann wird die erste Millisekunde dieses Quartals zurückgegeben, der 1. April um 12:00:00 AM.

Beispiel 3 – first_month_of_year

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript verwendet den gleichen Datensatz und das gleiche Szenario wie das erste Beispiel. In diesem Beispiel müssen wir aber den 1. März als Beginn des Geschäftsjahres festlegen.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        quarterstart(date,0,3) as start_of_quarter,
        timestamp(quarterstart(date,0,3)) as start_of_quarter_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
```

```
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `start_of_quarter`
- `start_of_quarter_timestamp`

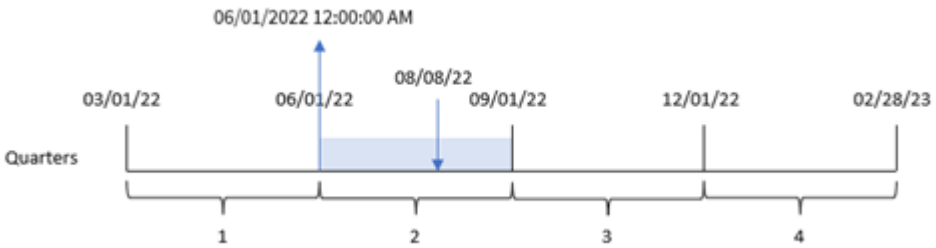
Ergebnistabelle

date	start_of_quarter	start_of_quarter_timestamp
1/7/2022	12/01/2021	12/1/2021 12:00:00 AM
1/19/2022	12/01/2021	12/1/2021 12:00:00 AM
2/5/2022	12/01/2021	12/1/2021 12:00:00 AM
2/28/2022	12/01/2021	12/1/2021 12:00:00 AM
3/16/2022	03/01/2022	3/1/2022 12:00:00 AM
4/1/2022	03/01/2022	3/1/2022 12:00:00 AM
5/7/2022	03/01/2022	3/1/2022 12:00:00 AM
5/16/2022	03/01/2022	3/1/2022 12:00:00 AM
6/15/2022	06/01/2022	6/1/2022 12:00:00 AM
6/26/2022	06/01/2022	6/1/2022 12:00:00 AM
7/9/2022	06/01/2022	6/1/2022 12:00:00 AM
7/22/2022	06/01/2022	6/1/2022 12:00:00 AM
7/23/2022	06/01/2022	6/1/2022 12:00:00 AM
7/27/2022	06/01/2022	6/1/2022 12:00:00 AM
8/2/2022	06/01/2022	6/1/2022 12:00:00 AM
8/8/2022	06/01/2022	6/1/2022 12:00:00 AM

date	start_of_quarter	start_of_quarter_timestamp
8/19/2022	06/01/2022	6/1/2022 12:00:00 AM
9/26/2022	09/01/2022	9/1/2022 12:00:00 AM
10/14/2022	09/01/2022	9/1/2022 12:00:00 AM
10/29/2022	09/01/2022	9/1/2022 12:00:00 AM

Da in diesem Fall das Argument `first_month_of_year` von 3 in der Funktion `quarterstart()` verwendet wird, wird der Start des Jahres vom 1. Januar auf den 1. März verschoben.

Diagramm der Funktion `quarterstart()`, Beispiel „first_month_of_year“



Transaktion 8203 fand am 8. August statt. Da der Anfang des Jahres der 1. März ist, sind die Quartale des Jahres März-Mai, Juni-August, September-November und Dezember-Februar. Die Funktion `quarterstart()` identifiziert, dass die Transaktion im Quartal zwischen Anfang Juni und Ende August stattfand und gibt die erste Millisekunde dieses Quartals zurück. Dies ist der 1. Juni um 12:00:00 AM.

Beispiel 4 – Diagrammobjektbeispiel

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript verwendet den gleichen Datensatz und das gleiche Szenario wie das erste Beispiel.

In diesem Beispiel wird jedoch der unveränderte Datensatz in die Anwendung geladen. Die Berechnung, die einen Zeitstempel für das Ende des Quartals zurückgibt, in dem die Transaktionen stattfanden, wird als Kennzahl in einem Diagrammobjekt der Anwendung erstellt.

Ladeskript

```
Transactions:
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
```

```
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu: `date`.

Fügen Sie die folgenden Kennzahlen hinzu:

- `=quarterstart(date)`
- `=timestamp(quarterstart(date))`

Ergebnistabelle

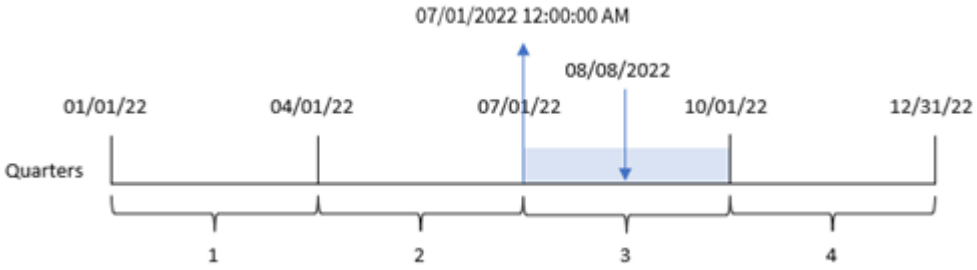
date	=quarterstart(date)	=timestamp(quarterstart(date))
10/14/2022	10/01/2022	10/1/2022 12:00:00 AM
10/29/2022	10/01/2022	10/1/2022 12:00:00 AM
7/9/2022	07/01/2022	7/1/2022 12:00:00 AM
7/22/2022	07/01/2022	7/1/2022 12:00:00 AM
7/23/2022	07/01/2022	7/1/2022 12:00:00 AM
7/27/2022	07/01/2022	7/1/2022 12:00:00 AM
8/2/2022	07/01/2022	7/1/2022 12:00:00 AM
8/8/2022	07/01/2022	7/1/2022 12:00:00 AM
8/19/2022	07/01/2022	7/1/2022 12:00:00 AM
9/26/2022	07/01/2022	7/1/2022 12:00:00 AM
4/1/2022	04/01/2022	4/1/2022 12:00:00 AM

date	=quarterstart(date)	=timestamp(quarterstart(date))
5/7/2022	04/01/2022	4/1/2022 12:00:00 AM
5/16/2022	04/01/2022	4/1/2022 12:00:00 AM
6/15/2022	04/01/2022	4/1/2022 12:00:00 AM
6/26/2022	04/01/2022	4/1/2022 12:00:00 AM
1/7/2022	01/01/2022	1/1/2022 12:00:00 AM
1/19/2022	01/01/2022	1/1/2022 12:00:00 AM
2/5/2022	01/01/2022	1/1/2022 12:00:00 AM
2/28/2022	01/01/2022	1/1/2022 12:00:00 AM
3/16/2022	01/01/2022	1/1/2022 12:00:00 AM

Die Kennzahl `start_of_quarter` wird im Diagrammobjekt erstellt, indem die Funktion `quarterstart()` verwendet und das Feld `date` als Argument der Funktion übergeben wird.

Die Funktion `quarterstart()` identifiziert, in welches Quartal der Datumswert fällt, und gibt einen Zeitstempel für die erste Millisekunde dieses Quartals zurück.

Diagramm der Funktion `quarterstart()`, Diagrammobjektbeispiel



Transaktion 8203 fand am 8. August statt. Die Funktion `quarterstart()` identifiziert, dass die Transaktion im dritten Quartal stattfand und gibt die erste Millisekunde dieses Quartals zurück. Dieser zurückgegebene Wert ist der 1. Juli um 12:00:00 AM.

Beispiel 5 – Szenario

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz mit einer Reihe von Darlehenssalden wird in eine Tabelle namens `Loans` geladen.
- Die Daten bestehen aus der Darlehens-ID, dem Saldo zum Quartalsbeginn und dem einfachen Zinssatz, der für jedes Darlehen pro Jahr berechnet wird.

Der Endbenutzer möchte ein Diagrammobjekt, das nach Darlehens-ID die aktuellen Zinsen anzeigt, die für jedes Darlehen im Quartal bis dato aufgelaufen sind.

Ladeskript

```
Loans:
Load
*
Inline
[
loan_id,start_balance,rate
8188,$10000.00,0.024
8189,$15000.00,0.057
8190,$17500.00,0.024
8191,$21000.00,0.034
8192,$90000.00,0.084
];
```

Ergebnisse

Gehen Sie folgendermaßen vor:

1. Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:
   - `loan_id`
   - `start_balance`
2. Erstellen Sie dann die folgende Kennzahl, um die kumulierten Zinsen zu berechnen: `=start_balance*(rate*(today(1)-quarterstart(today(1)))/365)`
3. Legen Sie das **Zahlenformat** der Kennzahl auf **Währung** fest.

Ergebnistabelle

loan_id	start_balance	=start_balance*(rate*(today(1)-quarterstart(today(1)))/365)
8188	$10000.00	$15.07
8189	$15000.00	$128.84
8190	$17500.00	$63.29
8191	$21000.00	$107.59
8192	$90000.00	$1139.18

Die Funktion `quarterstart()` verwendet das aktuelle Datum als einziges Argument und gibt das Startdatum des aktuellen Jahres zurück. Die Formel zieht dieses Ergebnis vom aktuellen Datum ab und gibt die Anzahl der Tage zurück, die bisher im Quartal verstrichen sind.

Dieser Wert wird dann mit dem Zinssatz multipliziert und durch 365 geteilt, um den effektiven Zinssatz für diesen Zeitraum zurückzugeben. Das Ergebnis wird dann mit dem Anfangssaldo des Darlehens multipliziert, was die Zinsen ergibt, die bislang in diesem Quartal aufgelaufen sind.

second

Diese Funktion liefert die Sekunden als ganze Zahl, wenn **expression** entsprechend dem Standardformat als Uhrzeit interpretiert wird.

Syntax:

```
second (expression)
```

Rückgabe Datentyp: ganze Zahl

Verwendung

Die Funktion `second()` ist nützlich, wenn Sie Aggregierungen nach Sekunde vergleichen möchten. Sie können die Funktion beispielsweise verwenden, wenn Sie die Verteilung der Anzahl der Aktivitäten nach Sekunde anzeigen möchten.

Diese Dimensionen können entweder im Ladeskript erstellt werden, indem die Funktion zum Erstellen eines Felds in einer Master-Kalender-Tabelle verwendet wird, oder sie können direkt in einem Diagramm als berechnete Dimensionenç verwendet werden.

Funktionsbeispiele

Beispiel	Ergebnis
`second( '09:14:36' )`	gibt 36 zurück
`second( '0.5555' )`	gibt 55 zurück (da 0,5555 = 13:19:55)

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Beispiel 1 – Variable

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz, der Transaktionen nach Zeitstempel enthält, wird in eine Tabelle namens `Transactions` geladen.
- Die Standardsystemvariable `TimeStamp` (`M/D/YYYY h:mm:ss[.fff] TT`) wird verwendet.
- Es wird ein Feld `second` erstellt, um zu berechnen, wann Einkäufe stattfanden.

Ladeskript

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Transactions:
    Load
        *,
        second(date) as second
        ;
Load
*
Inline
[
id,date,amount
9497,'01/05/2022 7:04:57 PM',47.25
9498,'01/03/2022 2:21:53 PM',51.75
9499,'01/03/2022 5:40:49 AM',73.53
9500,'01/04/2022 6:49:38 PM',15.35
9501,'01/01/2022 10:10:22 PM',31.43
9502,'01/05/2022 7:34:46 PM',13.24
9503,'01/06/2022 10:58:34 PM',74.34
9504,'01/06/2022 11:29:38 AM',50.00
9505,'01/02/2022 8:35:54 AM',36.34
9506,'01/06/2022 8:49:09 AM',74.23
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `second`

Ergebnistabelle

date	second
01/01/2022 10:10:22 PM	22
01/02/2022 8:35:54 AM	54
01/03/2022 5:40:49 AM	49
01/03/2022 2:21:53 PM	53
01/04/2022 6:49:38 PM	38
01/05/2022 7:04:57 PM	57
01/05/2022 7:34:46 PM	46
01/06/2022 8:49:09 AM	9
01/06/2022 11:29:38 AM	38
01/06/2022 10:58:34 PM	34

Die Werte im Feld `second` werden anhand der Funktion `second()` erstellt und übergeben das Datum als die Formel im vorangehenden load-Befehl.

Beispiel 2 – Diagrammobjekt

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript verwendet den gleichen Datensatz und das gleiche Szenario wie das erste Beispiel. In diesem Beispiel wird jedoch der unveränderte Datensatz in die Anwendung geladen. Die Werte für `second` werden anhand einer Kennzahl in einem Diagrammobjekt berechnet.

Ladeskript

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

Transactions:
Load
*
Inline
[
id,date,amount
9497,'01/05/2022 7:04:57 PM',47.25
9498,'01/03/2022 2:21:53 PM',51.75
9499,'01/03/2022 5:40:49 AM',73.53
9500,'01/04/2022 6:49:38 PM',15.35
9501,'01/01/2022 10:10:22 PM',31.43
9502,'01/05/2022 7:34:46 PM',13.24
9503,'01/06/2022 10:58:34 PM',74.34
```

```
9504,'01/06/2022 11:29:38 AM',50.00
9505,'01/02/2022 8:35:54 AM',36.34
9506,'01/06/2022 8:49:09 AM',74.23
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu:`date`.

Erstellen Sie die folgende Kennzahl:

```
=second(date)
```

Ergebnistabelle

date	=second(date)
01/01/2022 10:10:22 PM	22
01/02/2022 8:35:54 AM	54
01/03/2022 5:40:49 AM	49
01/03/2022 2:21:53 PM	53
01/04/2022 6:49:38 PM	38
01/05/2022 7:04:57 PM	57
01/05/2022 7:34:46 PM	46
01/06/2022 8:49:09 AM	9
01/06/2022 11:29:38 AM	38
01/06/2022 10:58:34 PM	34

Die Werte für `second` werden erstellt, indem die Funktion `second()` verwendet und das Datum als Formel in einer Kennzahl für das Diagrammobjekt übergeben wird.

Beispiel 3 – Szenario

Ladeskript und Diagrammformeln

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz mit Zeitstempeln wird generiert, um den Datenverkehr auf der Website für den Eintrittskartenverkauf für ein Festival darzustellen. Diese Zeitstempel und eine zugehörige `id` werden in eine Tabelle namens `Web_Traffic` geladen.
- Die `TimeStamp`-Systemvariable `M/D/YYYY h:mm:ss[.fff] TT` wird verwendet.

In diesem Szenario gab es 10.000 Eintrittskarten, die ab 9:00 AM am 20. Mai 2021 zum Verkauf angeboten wurden. Eine Minute später waren die Eintrittskarten ausverkauft.

Der Benutzer möchte ein Diagrammobjekt, das die Anzahl der Websitebesuche nach Sekunden zeigt.

Ladeskript

```
SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';

tmpTimeStampCreator:
load
    makedate(2022,05,20) as date
AutoGenerate 1;

join load
    maketime(9+floor(rand()*2),0,floor(rand()*59)) as time
autogenerate 10000;

Web_Traffic:
load
    recno() as id,
    timestamp(date + time) as timestamp
resident tmpTimeStampCreator;

drop table tmpTimeStampCreator;
```

Ergebnisse

Gehen Sie folgendermaßen vor:

1. Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle.
2. Erstellen Sie dann eine berechnete Dimension anhand der folgenden Formel: `=second(timestamp)`
3. Erstellen Sie eine Aggregierungskennzahl, um die Gesamtzahl der Zugriffe berechnen: `=count(id)`

Die Ergebnistabelle gleicht der folgenden Tabelle, aber mit anderen Werten für die Aggregierungskennzahl:

Ergebnistabelle

second(timestamp)	=count(id)
0	150
1	184
2	163
3	178
4	179
5	158

second(timestamp)	=count(id)
6	177
7	169
8	149
9	186
10	169
11	179
12	186
13	182
14	180
15	153
16	191
17	203
18	158
19	159
20	163
+ 39 weitere Zeilen	

setdateyear

Als Eingabe verwendet diese Funktion einen **timestamp** und ein **year** und aktualisiert den **timestamp** mit dem in der Eingabe festgelegten **year** .

Syntax:

```
setdateyear (timestamp, year)
```

Rückgabe Datentyp: dual

Argumente:

Argumente

Argument	Beschreibung
timestamp	Ein Standardzeitstempel in Qlik Sense (es kann auch nur ein Datum sein).
year	Eine vierstellige Jahreszahl.

Beispiele und Ergebnisse:
In diesen Beispielen wird das Datumsformat **DD/MM/YYYY** verwendet. Das Datumsformat wird im Befehl **SET DateFormat** oben in Ihrem Datenladeskript angegeben. Ändern Sie das Format in den Beispielen nach Bedarf.

Skriptbeispiele

Beispiel	Ergebnis
`setdateyear ('29/10/2005', 2013)`	Liefert '29/10/2013
`setdateyear ('29/10/2005 04:26:14', 2013)`	Liefert '29/10/2013 04:26:14' Damit der Zeitanteil des Zeitstempels in einer Visualisierung angezeigt wird, müssen Sie Datum als Zahlenformat und einen Wert für die Formatierung zur Anzeige von Zeitwerten auswählen.

Beispiel:

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Fügen Sie einem Arbeitsblatt in Ihrer App dann die Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um das Ergebnis anzuzeigen.

```
SetYear:

Load *,

SetDateYear(testdates, 2013) as NewYear

Inline [

testdates

1/11/2012

10/12/2012

1/5/2013

2/1/2013

19/5/2013

15/9/2013

11/12/2013

2/3/2014

14/5/2014

13/6/2014

7/7/2014
```

```
4/8/2014

];
```

Die sich daraus ergebende Tabelle enthält die ursprünglichen Daten sowie eine Spalte, in der das Jahr auf 2013 festgelegt wurde.

Ergebnistabelle

testdates	NewYear
1/11/2012	1/11/2013
10/12/2012	10/12/2013
2/1/2012	2/1/2013
1/5/2013	1/5/2013
19/5/2013	19/5/2013
15/9/2013	15/9/2013
11/12/2013	11/12/2013
2/3/2014	2/3/2013
14/5/2014	14/5/2013
13/6/2014	13/6/2013
7/7/2014	7/7/2013
4/8/2014	4/8/2013

setdateyearmonth

Als Eingabe verwendet diese Funktion einen **timestamp**, einen **month** und ein **year** und aktualisiert den **timestamp** mit dem in der Eingabe festgelegten **year** und dem **month** . .

Syntax:

```
SetDateYearMonth (timestamp, year, month)
```

Rückgabe Datentyp: dual

Argumente:

Argumente

Argument	Beschreibung
timestamp	Ein Standardzeitstempel in Qlik Sense (es kann auch nur ein Datum sein).
year	Eine vierstellige Jahreszahl.
month	Eine ein- oder zweistellige Monatszahl.

Beispiele und Ergebnisse:
In diesen Beispielen wird das Datumsformat **DD/MM/YYYY** verwendet. Das Datumsformat wird im Befehl **SET DateFormat** oben in Ihrem Datenladeskript angegeben. Ändern Sie das Format in den Beispielen nach Bedarf.

Skriptbeispiele

Beispiel	Ergebnis
`setdateyearmonth ('29/10/2005', 2013, 3)`	Liefert '29/03/2013
`setdateyearmonth ('29/10/2005 04:26:14', 2013, 3)`	Liefert '29/03/2013 04:26:14' Damit der Zeitanteil des Zeitstempels in einer Visualisierung angezeigt wird, müssen Sie Datum als Zahlenformat und einen Wert für die Formatierung zur Anzeige von Zeitwerten auswählen.

Beispiel:

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Fügen Sie einem Arbeitsblatt in Ihrer App dann die Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um das Ergebnis anzuzeigen.

```
SetYearMonth:

Load *,

SetDateYearMonth(testdates, 2013,3) as NewYearMonth

Inline [

testdates

1/11/2012

10/12/2012

2/1/2013

19/5/2013

15/9/2013

11/12/2013

14/5/2014

13/6/2014

7/7/2014

4/8/2014

];
```

Die sich daraus ergebende Tabelle enthält die ursprünglichen Daten sowie eine Spalte, in der das Jahr auf 2013 festgelegt wurde.

Ergebnistabelle

testdates	NewYearMonth
1/11/2012	1/3/2013
10/12/2012	10/3/2013
2/1/2012	2/3/2013
19/5/2013	19/3/2013
15/9/2013	15/3/2013
11/12/2013	11/3/2013
14/5/2014	14/3/2013
13/6/2014	13/3/2013
7/7/2014	7/3/2013
4/8/2014	4/3/2013

timezone

Diese Funktion gibt die Zeitzone zurück, die für den Computer definiert ist, auf dem die Qlik-Engine ausgeführt wird.

Syntax:

```
TimeZone( )
```

Rückgabe Datentyp: dual

Beispiel:

`timezone( )`
Wenn Sie in einer Kennzahl Ihrer App eine andere Zeitzone anzeigen möchten, können Sie die Funktion `localtime()` in einer Kennzahl verwenden.

today

Diese Funktion gibt das aktuelle Datum zurück. Die Funktion gibt Werte im Systemvariablenformat `DateFormat` zurück.

Syntax:

```
today([ timer_mode])
```

Rückgabe Datentyp: dual

Die Funktion `today()` kann entweder im Ladeskript oder in Diagrammobjekten verwendet werden.

Der Standardwert von `timer_mode` ist 1.

Argumente

Argument	Beschreibung
timer_mode	Kann folgende Werte haben: 0 (Tag des letzten abgeschlossenen Datenladevorgangs) 1 (Tag des Funktionsaufrufs) 2 (Tag beim Öffnen der App) *Wenn Sie die Funktion in einem Ladeskript verwenden, gibt **timer_mode=0** den Tag des letzten abgeschlossenen Datenladevorgangs an, während **timer_mode=1** den Tag des aktuellen Datenladevorgangs angibt.*

Funktionsbeispiele

timer_mode value	Ergebnis bei Verwendung im Ladeskript	Ergebnis bei Verwendung im Diagrammobjekt
0	Gibt ein Datum im Format der Systemvariablen `DateFormat` für den letzten erfolgreichen Datenladevorgang vor dem neuesten Datenladevorgang zurück.	Gibt ein Datum im Format der Systemvariablen `DateFormat` für den neuesten Datenladevorgang zurück.
1	Gibt ein Datum im Format der Systemvariablen `DateFormat` für den neuesten Datenladevorgang zurück.	Gibt ein Datum im Format der Systemvariablen `DateFormat` für den Funktionsaufruf zurück.
2	Gibt ein Datum im Format der Systemvariablen `DateFormat` für den Zeitpunkt zurück, zu dem die Sitzung des Benutzers in der Anwendung begann. Dieser wird erst aktualisiert, wenn der Benutzer das Skript neu lädt.	Gibt das Datum im Format der Systemvariablen `DateFormat` für den Zeitpunkt zurück, zu dem die Sitzung des Benutzers in der Anwendung begann. Dieser wird aktualisiert, wenn eine neue Sitzung beginnt oder wenn die Daten in der Anwendung neu geladen werden.

Verwendung

Die Funktion `today()` wird häufig als Komponente innerhalb einer Formel verwendet. Zum Beispiel kann sie verwendet werden, um die Zinsen zu berechnen, die in einem Monat bis dato aufgelaufen sind.

Die folgende Tabelle erläutert das Ergebnis, das von der Funktion `today()` bei verschiedenen Werten für das Argument `timer_mode` zurückgegeben wird:

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Beispiel 1 – Generierung von Objekten anhand des Ladeskripts

Ladeskript und Ergebnisse

Übersicht

Im folgenden Beispiel werden drei Variablen mit der Funktion `today()` erstellt. Jede Variable verwendet eine der Optionen `timer_mode`, um deren Auswirkung zu demonstrieren.

Damit der Zweck der Variablen demonstriert wird, laden Sie das Skript und laden Sie es nach 24 Stunden ein zweites Mal. Das führt dazu, dass die Variablen `today(0)` und `today(1)` unterschiedliche Werte zeigen und damit ihren Zweck demonstrieren.

Ladeskript

```
LET vPreviousDataLoad = today(0);
LET vCurrentDataLoad = today(1);
LET vApplicationOpened = today(2);
```

Ergebnisse

Erstellen Sie drei Textfelder anhand der Anleitung unten, sobald die Daten ein zweites Mal geladen wurden.

Erstellen Sie zunächst ein Textfeld für zuvor geladene Daten.

Gehen Sie folgendermaßen vor:

1. Erstellen Sie ein Textfeld mithilfe des Diagrammobjekts **Text und Bild**.
2. Fügen Sie dem Objekt die folgende Kennzahl hinzu:
   `=vPreviousDataLoad`
3. Wählen Sie unter **Darstellung** die Option **Show titles** aus und fügen Sie den Titel „Zeit des vorherigen Neuladevorgangs“ zum Objekt hinzu.

Erstellen Sie als Nächstes ein Textfeld für Daten, die derzeit geladen werden.

Gehen Sie folgendermaßen vor:

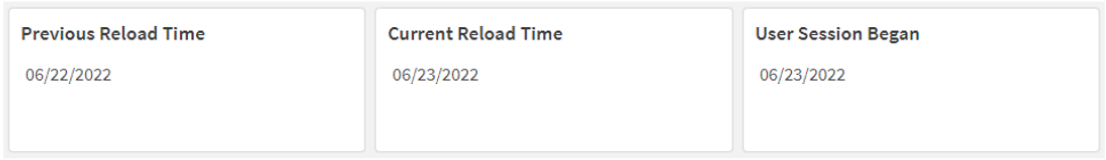
1. Erstellen Sie ein Textfeld mithilfe des Diagrammobjekts **Text und Bild**.
2. Fügen Sie dem Objekt die folgende Kennzahl hinzu:
   `=vCurrentDataLoad`
3. Wählen Sie unter **Darstellung** die Option **Show titles** aus und fügen Sie den Titel „Zeit des aktuellen Neuladevorgangs“ zum Objekt hinzu.

Erstellen Sie ein letztes Textfeld, das anzeigt, wann die Sitzung des Benutzers in der Anwendung gestartet wurde.

Gehen Sie folgendermaßen vor:

1. Erstellen Sie ein Textfeld mithilfe des Diagrammobjekts **Text und Bild**.
2. Fügen Sie dem Objekt die folgende Kennzahl hinzu:
   `=vApplicationOpened`
3. Wählen Sie unter **Darstellung** die Option **Show titles** aus und fügen Sie den Titel „Start der Benutzersitzung“ zum Objekt hinzu.

Diagramm mit Variablen, die anhand der Funktion `today()` im Ladeskript erstellt wurden



Die obige Abbildung zeigt Beispielwerte für jede der erstellten Variablen. Die Werte können beispielsweise wie folgt aussehen:

- Vorherige Ladezeit: 06/22/2022
- Aktuelle Ladezeit: 06/23/2022
- Beginn der Benutzersitzung: 06/23/2022

Beispiel 2 – Generierung von Objekten ohne Ladeskript

Ladeskript und Diagrammformel

Übersicht

Im folgenden Beispiel werden drei Diagrammobjekte mit der Funktion `today()` erstellt. Jedes Diagrammobjekt verwendet eine der Optionen für `timer_mode`, um deren Auswirkung zu demonstrieren.

Für dieses Beispiel gibt es kein Ladeskript.

Ergebnisse

Nachdem die Daten ein zweites Mal geladen wurden, erstellen Sie drei Textfelder.

Erstellen Sie zuerst ein Textfeld für den neuesten Datenladevorgang.

Gehen Sie folgendermaßen vor:

1. Erstellen Sie ein Textfeld mithilfe des Diagrammobjekts **Text und Bild**.
2. Fügen Sie die folgende Kennzahl hinzu.
   `=today(0)`
3. Wählen Sie unter **Darstellung** die Option **Titel anzeigen** aus und fügen Sie den Titel „Neuester Datenladevorgang“ zum Objekt hinzu.

Erstellen Sie dann ein Textfeld, das die aktuelle Uhrzeit anzeigt.

Gehen Sie folgendermaßen vor:

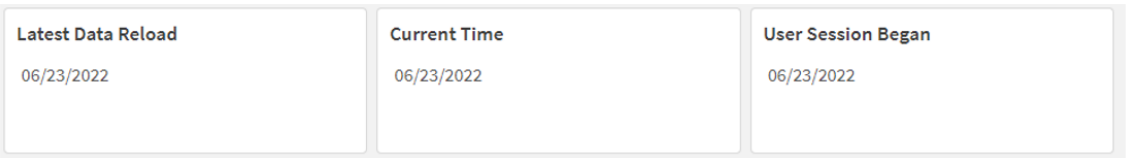
1. Erstellen Sie ein Textfeld mithilfe des Diagrammobjekts **Text und Bild**.
2. Fügen Sie die folgende Kennzahl hinzu:
   `=today(1)`
3. Wählen Sie unter **Darstellung** die Option**Titel anzeigen** aus und fügen Sie den Titel „Aktuelle Uhrzeit“ zum Objekt hinzu.

Erstellen Sie ein letztes Textfeld, das anzeigt, wann die Sitzung des Benutzers in der Anwendung gestartet wurde.

Gehen Sie folgendermaßen vor:

1. Erstellen Sie ein Textfeld mithilfe des Diagrammobjekts **Text und Bild**.
2. Fügen Sie die folgende Kennzahl hinzu:
   `=today(2)`
3. Wählen Sie unter **Darstellung** die Option **Titel anzeigen** aus und fügen Sie den Titel „Beginn der Benutzersitzung“ zum Objekt hinzu.

Diagramm mit Objekten, die anhand der Funktion `today()` ohne Ladeskript erstellt wurden



Die obige Abbildung zeigt Beispielwerte für jedes der erstellten Objekte. Die Werte können beispielsweise wie folgt aussehen:

- Neuester Datenladevorgang: 06/23/2022
- Aktuelle Uhrzeit: 06/23/2022
- Beginn der Benutzersitzung: 06/23/2022

Das Diagrammobjekt „Neuester Datenladevorgang“ verwendet einen Wert für `timer_mode` von 0. Damit wird ein Zeitstempel für die Uhrzeit zurückgegeben, zu der die Daten zum letzten Mal erfolgreich geladen wurden.

Das Diagrammobjekt „Aktuelle Uhrzeit“ verwendet einen Wert für `timer_mode` von 1. Damit wird die aktuelle Uhrzeit der Systemuhr zurückgegeben. Wenn das Arbeitsblatt oder Objekt aktualisiert wird, wird dieser Wert aktualisiert.

Das Diagrammobjekt „Beginn der Benutzersitzung“ verwendet einen Wert für `timer_mode` von 2. Damit wird der Zeitstempel des Zeitpunkts zurückgegeben, zu dem die Anwendung geöffnet wurde und die Benutzersitzung begann.

Beispiel 3 – Szenario

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz mit einer Reihe von Darlehenssalden wird in eine Tabelle namens `Loans` geladen.
- Tabellendaten mit Feldern für die Darlehens-ID, den Saldo zum Monatsbeginn und den einfachen Zinssatz, der für jedes Darlehen pro Jahr berechnet wird.

Der Endbenutzer möchte ein Diagrammobjekt, das nach Darlehens-ID die aktuellen Zinsen anzeigt, die für jedes Darlehen im Monat bis dato aufgelaufen sind. Die Anwendung wird zwar nur einmal pro Woche geladen, aber der Benutzer möchte, dass die Ergebnisse aktualisiert werden, sooft das Objekt oder die Anwendung aktualisiert werden.

Ladeskript

```
Loans:
Load
*
Inline
[
loan_id,start_balance,rate
8188,$10000.00,0.024
8189,$15000.00,0.057
8190,$17500.00,0.024
8191,$21000.00,0.034
8192,$90000.00,0.084
];
```

Ergebnisse

Gehen Sie folgendermaßen vor:

1. Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle.
2. Fügen Sie die folgenden Felder als Dimensionen hinzu:
   - `loan_id`
   - `start_balance`
3. Erstellen Sie dann eine Kennzahl, um die kumulierten Zinsen zu berechnen:
   `=start_balance*(rate*(today(1)-monthstart(today(1)))/365)`
4. Legen Sie das **Zahlenformat** der Kennzahl auf **Währung** fest.

Ergebnistabelle

loan_id	start_balance	=start_balance*(rate*(today(1)-monthstart(today(1)))/365)
8188	$10000.00	$16.44
8189	$15000.00	$58.56
8190	$17500.00	$28.77
8191	$21000.00	$48.90
8192	$90000.00	$517.81

Die Funktion `monthstart()` verwendet die Funktion `today()`, um das aktuelle Datum als einziges Argument zurückzugeben, und gibt das Startdatum des aktuellen Monats zurück. Die Formel zieht dieses Ergebnis vom aktuellen Datum ab, wobei erneut die Funktion `today()` verwendet wird, und gibt die Anzahl der Tage zurück, die bisher im Monat verstrichen sind.

Dieser Wert wird dann mit dem Zinssatz multipliziert und durch 365 geteilt, um den effektiven Zinssatz für diesen Zeitraum zurückzugeben. Das Ergebnis wird dann mit dem Anfangssaldo des Darlehens multipliziert, was die Zinsen ergibt, die bislang in diesem Monat aufgelaufen sind.

Da der Wert von 1 als Argument `timer_mode` in den Funktionen `today()` innerhalb der Formel verwendet wird, wird bei jeder Aktualisierung des Diagrammobjekts (z. B. durch Öffnen der Anwendung, Aktualisieren der Seite, Wechseln zwischen Arbeitsblättern usw.) das aktuelle Datum zurückgegeben, und die Ergebnisse werden entsprechend aktualisiert.

UTC

Liefert die aktuelle Coordinated Universal Time.

Syntax:

```
UTC( )
```

Rückgabe Datentyp: dual

Beispiel:

```
utc( )
```

week

Diese Funktion gibt die Wochennummer zurück, die dem eingegebenen Datum entspricht.

Syntax:

```
week(timestamp [, first_week_day [, broken_weeks [, reference_day]]])
```

Rückgabe Datentyp: ganze Zahl

Argumente

Argument	Beschreibung
timestamp	Datum oder Zeitstempel für die Evaluierung.
first_week_day	Legt den Tag fest, an dem die Woche beginnt. Ist nichts definiert, wird der Wert der Variablen **FirstWeekDay** verwendet. Die möglichen Werte für **first_week_day** sind 0 für Montag, 1 für Dienstag, 2 für Mittwoch, 3 für Donnerstag, 4 für Freitag, 5 für Samstag und 6 für Sonntag. Weitere Informationen über die Systemvariable finden Sie unter *FirstWeekDay (page 233)*.
broken_weeks	Wenn Sie **broken_weeks** nicht angeben, wird der Wert der Variablen **BrokenWeeks** verwendet, um festzulegen, ob die Wochen gestückelt sind oder nicht.
reference_day	Wenn Sie **reference_day** nicht angeben, wird der Wert der Variablen **ReferenceDay** verwendet, um festzulegen, welcher Tag im Januar als Referenztag für die Definition von Woche 1 konfiguriert wird. Standardmäßig verwenden Qlik Sense-Funktionen 4 als Referenztag. Das heißt, dass die Woche 1 den 4. Januar enthalten muss, oder anders ausgedrückt: Woche 1 muss immer mindestens 4 Tage im Januar enthalten.

Die Funktion `week()` bestimmt, in welche Woche das Datum fällt, und gibt die Wochennummer zurück.

In Qlik Sense werden die regionalen Einstellungen bei der Erstellung der App abgerufen, und die entsprechenden Einstellungen werden im Skript als Umgebungsvariablen gespeichert. Sie werden zur Bestimmung der Wochennummer verwendet.

Das bedeutet, dass die meisten europäischen App-Entwickler die folgenden Umgebungsvariablen gemäß der ISO 8601-Definition erhalten:

```
Set FirstWeekDay =0;    // Monday as first week day
Set BrokenWeeks  =0;    // Use unbroken weeks
Set ReferenceDay =4;    // Jan 4th is always in week 1
```

Ein nordamerikanischer App-Entwickler erhält oft die folgenden Umgebungsvariablen:

```
Set FirstWeekDay =6;    // Sunday as first week day
Set BrokenWeeks  =1;    // Use broken weeks
Set ReferenceDay =1;    // Jan 1st is always in week 1
```

Der erste Tag der Woche wird durch die Systemvariable `FirstWeekDay` bestimmt. Sie können auch den ersten Tag der Woche ändern, indem Sie das Argument `first_week_day` in der Funktion `week()` verwenden.

Wenn Ihre Anwendung gestückelte Wochen verwendet, beginnt die Zählung der Wochennummer am 1. Januar und endet am Tag vor der Systemvariablen `FirstWeekDay`, unabhängig davon, wie viele Tage verstrichen sind.

Wenn Ihre Anwendung vollständige Wochen verwendet, kann Woche 1 im Vorjahr oder in den ersten Tagen im Januar beginnen. Das hängt davon ab, wie Sie die Umgebungsvariablen `FirstWeekDay` und `ReferenceDay` verwenden.

Verwendung

Die Funktion `The week()` ist nützlich, wenn Sie Aggregierungen nach Wochen vergleichen möchten. Sie kann beispielsweise verwendet werden, wenn Sie den Gesamtumsatz von Produkten nach Woche anzeigen möchten. Die Funktion `week()` wird anstelle von `weekname()` gewählt, wenn der Benutzer möchte, dass die Berechnung nicht notwendigerweise die Systemvariablen `BrokenWeeks`, `FirstWeekDay` oder `ReferenceDay` der Anwendung verwendet.

Das ist beispielsweise der Fall, wenn Sie den Gesamtumsatz von Produkten nach Woche anzeigen möchten.

Wenn die Anwendung vollständige Wochen verwendet, kann Woche 1 Tage vom Dezember des Vorjahres enthalten oder Tage im Januar des laufenden Jahres ausschließen. Wenn die Anwendung gestückelte Wochen verwendet, kann Woche 1 weniger als sieben Tage enthalten.

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Die folgenden Beispiele nehmen an

```
Set DateFormat= 'MM/DD/YYYY';
Set FirstWeekDay=0;
Set BrokenWeeks=0;
Set ReferenceDay=4;
```

Funktionsbeispiele

Beispiel	Ergebnis
week('12/28/2021')	Gibt 52 zurück.
week(44614)	Gibt 8 zurück, da dies die Seriennummer für 02/22/2022 ist.
week('01/03/2021')	Gibt 53 zurück.
week('01/03/2021',6)	Liefert 1.

Beispiel 1 – Standardsystemvariablen

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz, der eine Reihe von Transaktionen für die letzte Woche in 2021 und die ersten beiden Wochen in 2022 enthält, wird in eine Tabelle namens `Transactions` geladen.
- Datumsfeld, das im Format der Systemvariablen `DateFormat` (MM/TT/JJJJ) bereitgestellt wird
- Erstellung eines Felds `week_number`, das das Jahr und die Nummer der Woche, in der die Transaktionen stattfanden, zurückgibt
- Erstellung eines Felds namens `week_day`, das den Wochentagswert für jedes Transaktionsdatum zurückgibt

Ladeskript

```
SET DateFormat='MM/DD/YYYY';
SET FirstWeekDay=6;
SET BrokenWeeks=1;
SET ReferenceDay=0;

Transactions:
    Load
        *,
        WeekDay(date) as week_day,
        Week(date) as week_number
        ;
Load
*
Inline
[
id,date,amount
8183,12/27/2021,58.27
8184,12/28/2021,67.42
8185,12/29/2021,23.80
8186,12/30/2021,82.06
8187,12/31/2021,40.56
```

```
8188,01/01/2022,37.23
8189,01/02/2022,17.17
8190,01/03/2022,88.27
8191,01/04/2022,57.42
8192,01/05/2022,53.80
8193,01/06/2022,82.06
8194,01/07/2022,40.56
8195,01/08/2022,53.67
8196,01/09/2022,26.63
8197,01/10/2022,72.48
8198,01/11/2022,18.37
8199,01/12/2022,45.26
8200,01/13/2022,58.23
8201,01/14/2022,18.52
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `id`
- `date`
- `week_day`
- `week_number`

Ergebnistabelle

id	date	week_day	week_number
8183	12/27/2021	Mo	53
8184	12/28/2021	Di	53
8185	12/29/2021	Mi	53
8186	12/30/2021	Do	53
8187	12/31/2021	Fr	53
8188	01/01/2022	Sa	1
8189	01/02/2022	So	2
8190	01/03/2022	Mo	2
8191	01/04/2022	Di	2
8192	01/05/2022	Mi	2
8193	01/06/2022	Do	2
8194	01/07/2022	Fr	2
8195	01/08/2022	Sa	2

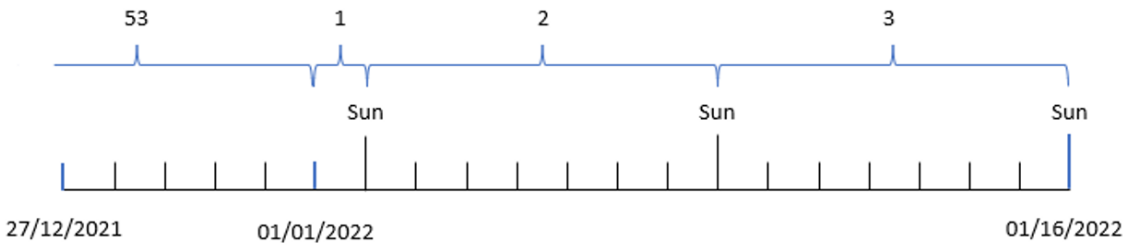
id	date	week_day	week_number
8196	01/09/2022	So	3
8197	01/10/2022	Mo	3
8198	01/11/2022	Di	3
8199	01/12/2022	Mi	3
8200	01/13/2022	Do	3
8201	01/14/2022	Fr	3

Das Feld `week_number` wird im vorangehenden load-Befehl erstellt, indem die Funktion `week()` verwendet und das Feld `date` als Argument der Funktion übergeben wird.

Es werden keine anderen Parameter an die Funktion übergeben, und daher gelten die folgenden, die Funktion `week()` betreffenden Standardvariablen:

- `BrokenWeeks`: Die Wochenzählung beginnt am 1. Januar.
- `FirstWeekDay`: Der erste Tag der Woche ist Sonntag.

Diagramm der Funktion `week()` mit Verwendung der Standardsystemvariablen



Da die Anwendung die Standardsystemvariable `BrokenWeeks` verwendet, beginnt Woche 1 am 1. Januar, einem Samstag.

Aufgrund der Standardsystemvariablen `FirstWeekDay` beginnen die Wochen an einem Sonntag. Der erste Sonntag nach dem 1. Januar ist der 2. Januar. Also beginnt an diesem Tag Woche 2.

Beispiel 2 – first_week_day

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Es wird ein Feld `week_number` erstellt, das das Jahr und die Nummer der Woche, in der die Transaktionen stattfanden, zurückgibt.
- Es wird ein Feld namens `week_day` erstellt, dass den Wochentagswert für jedes Transaktionsdatum zurückgibt.

In diesem Beispiel soll der Start der Woche auf Dienstag festgelegt werden.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';
SET FirstWeekDay=6;
SET BrokenWeeks=1;
SET ReferenceDay=0;

Transactions:
    Load
        *,
        WeekDay(date) as week_day,
        Week(date,1) as week_number
        ;
Load
*
Inline
[
id,date,amount
8183,12/27/2022,58.27
8184,12/28/2022,67.42
8185,12/29/2022,23.80
8186,12/30/2022,82.06
8187,12/31/2021,40.56
8188,01/01/2022,37.23
8189,01/02/2022,17.17
8190,01/03/2022,88.27
8191,01/04/2022,57.42
8192,01/05/2022,53.80
8193,01/06/2022,82.06
8194,01/07/2022,40.56
8195,01/08/2022,53.67
8196,01/09/2022,26.63
8197,01/10/2022,72.48
8198,01/11/2022,18.37
8199,01/12/2022,45.26
8200,01/13/2022,58.23
8201,01/14/2022,18.52
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

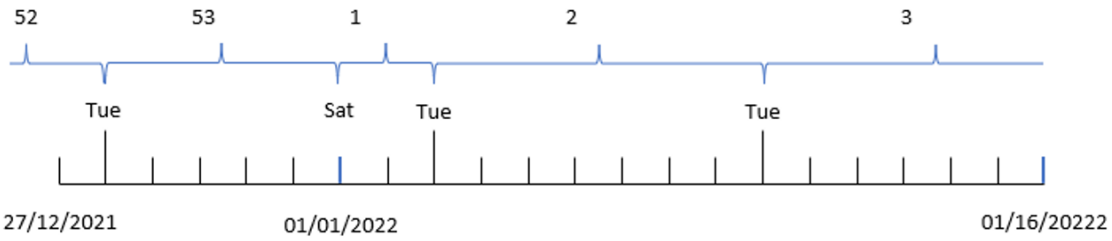
- id
- date
- week_day
- week_number

Ergebnistabelle

id	date	week_day	week_number
8183	12/27/2021	Mo	52
8184	12/28/2021	Di	53
8185	12/29/2021	Mi	53
8186	12/30/2021	Do	53
8187	12/31/2021	Fr	53
8188	01/01/2022	Sa	1
8189	01/02/2022	So	1
8190	01/03/2022	Mo	1
8191	01/04/2022	Di	2
8192	01/05/2022	Mi	2
8193	01/06/2022	Do	2
8194	01/07/2022	Fr	2
8195	01/08/2022	Sa	2
8196	01/09/2022	So	2
8197	01/10/2022	Mo	2
8198	01/11/2022	Di	3
8199	01/12/2022	Mi	3
8200	01/13/2022	Do	3
8201	01/14/2022	Fr	3

Die Anwendung verwendet immer noch gestückelte Wochen. Das Argument `first_week_day` wurde jedoch in der Funktion `week()` auf 1 festgelegt. Damit wird der erste Tag der Woche auf einen Dienstag festgelegt.

Diagramm der Funktion week(), Beispiel „first_week_day“



Die Anwendung verwendet die Standardsystemvariable `BrokenWeeks`, daher beginnt Woche 1 am 1. Januar, einem Samstag.

Das Argument `first_week_day` der Funktion `week()` legt den ersten Wochentag auf einen Dienstag fest. Daher beginnt Woche 53 am 28. Dezember 2021.

Da aber die Funktion immer noch gestückelte Wochen verwendet, ist Woche 1 nur zwei Tage lang, da der erste Dienstag nach dem 1. Januar am 3. Januar liegt.

Beispiel 3 – unbroken_weeks

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript verwendet den gleichen Datensatz und das gleiche Szenario wie das erste Beispiel.

In diesem Beispiel werden nicht gestückelte Wochen verwendet.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';
SET FirstWeekDay=6;
SET BrokenWeeks=1;
SET ReferenceDay=0;

Transactions:
    Load
        *,
        WeekDay(date) as week_day,
        Week(date,6,0) as week_number
        ;
Load
*
Inline
[
id,date,amount
8183,12/27/2022,58.27
```

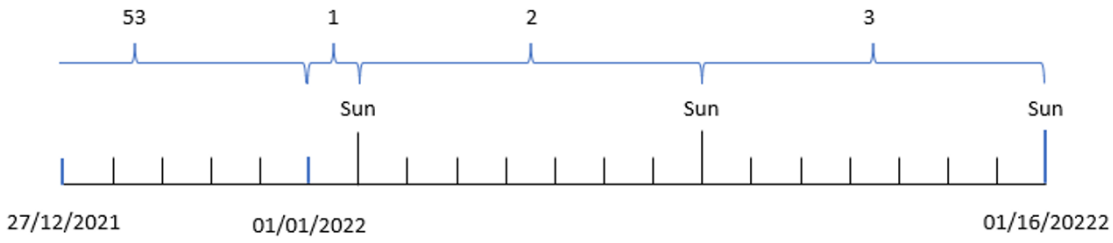
```
8184,12/28/2022,67.42
8185,12/29/2022,23.80
8186,12/30/2022,82.06
8187,12/31/2021,40.56
8188,01/01/2022,37.23
8189,01/02/2022,17.17
8190,01/03/2022,88.27
8191,01/04/2022,57.42
8192,01/05/2022,53.80
8193,01/06/2022,82.06
8194,01/07/2022,40.56
8195,01/08/2022,53.67
8196,01/09/2022,26.63
8197,01/10/2022,72.48
8198,01/11/2022,18.37
8199,01/12/2022,45.26
8200,01/13/2022,58.23
8201,01/14/2022,18.52
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `id`
- `date`
- `week_day`
- `week_number`

Diagramm der Funktion week(), Diagrammobjektbeispiel



Ergebnistabelle

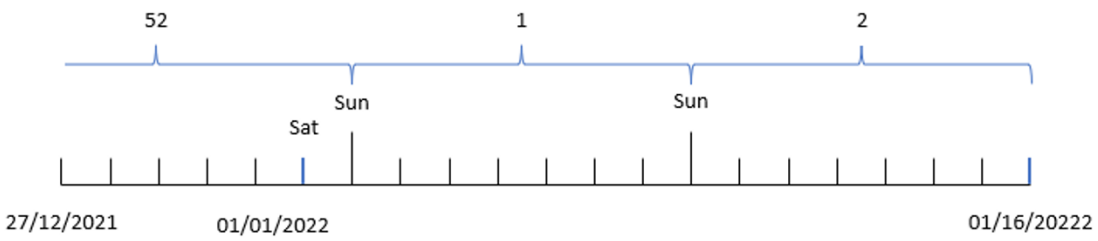
id	date	week_day	week_number
8183	12/27/2021	Mo	52
8184	12/28/2021	Di	52
8185	12/29/2021	Mi	52

id	date	week_day	week_number
8186	12/30/2021	Do	52
8187	12/31/2021	Fr	52
8188	01/01/2022	Sa	52
8189	01/02/2022	So	1
8190	01/03/2022	Mo	1
8191	01/04/2022	Di	1
8192	01/05/2022	Mi	1
8193	01/06/2022	Do	1
8194	01/07/2022	Fr	1
8195	01/08/2022	Sa	1
8196	01/09/2022	So	2
8197	01/10/2022	Mo	2
8198	01/11/2022	Di	2
8199	01/12/2022	Mi	2
8200	01/13/2022	Do	2
8201	01/14/2022	Fr	2

Der Parameter „first_week_date“ wird auf 1 festgelegt, wodurch Dienstag zum ersten Tag der Woche wird. Der Parameter „broken_weeks“ wird auf 0 festgelegt, was erzwingt, dass die Funktion nicht gestückelte Wochen verwendet. Im dritten Parameter wird abschließend „reference_day“ auf 2 festgelegt.

Der Parameter `first_week_date` wird auf 6 festgelegt, wodurch Sonntag zum ersten Tag der Woche wird. Der Parameter `broken_weeks` wird auf 0 festgelegt, was erzwingt, dass die Funktion nicht gestückelte Wochen verwendet.

Diagramm der Funktion „week()“, Beispiel mit nicht gestückelten Wochen



Wenn ungestückelte Wochen verwendet werden, beginnt Woche 1 nicht notwendigerweise am 1. Januar; es ist vielmehr erforderlich, dass sie mindestens vier Tage umfasst. Daher endet im Datensatz Woche 52 am Samstag, den 1. Januar 2022. Woche 1 beginnt dann an der Systemvariablen `FirstWeekDay`. Das ist Sonntag, der 2. Januar. Diese Woche endet am darauffolgenden Samstag, dem 8. Januar.

Beispiel 4 – reference_day

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Derselbe Datensatz und dasselbe Szenario wie im dritten Beispiel.
- Es wird ein Feld `week_number` erstellt, das das Jahr und die Nummer der Woche, in der die Transaktionen stattfanden, zurückgibt.
- Es wird ein Feld namens `week_day` erstellt, dass den Wochentagswert für jedes Transaktionsdatum zurückgibt.

Zusätzlich müssen die folgenden Bedingungen erfüllt sein:

- Die Arbeitswoche beginnt an einem Dienstag.
- Das Unternehmen verwendet nicht gestückelte Wochen.
- Der Wert für `reference_day` ist 2. Die Mindestanzahl Tage im Januar in Woche 1 beträgt also 2.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';
SET FirstWeekDay=6;
SET BrokenWeeks=1;
SET ReferenceDay=0;

Transactions:
    Load
        *,
        WeekDay(date) as week_day,
        Week(date,1,0,2) as week_number
        ;
Load
*
Inline
[
id,date,amount
8183,12/27/2022,58.27
8184,12/28/2022,67.42
8185,12/29/2022,23.80
8186,12/30/2022,82.06
8187,12/31/2021,40.56
8188,01/01/2022,37.23
```

```
8189,01/02/2022,17.17
8190,01/03/2022,88.27
8191,01/04/2022,57.42
8192,01/05/2022,53.80
8193,01/06/2022,82.06
8194,01/07/2022,40.56
8195,01/08/2022,53.67
8196,01/09/2022,26.63
8197,01/10/2022,72.48
8198,01/11/2022,18.37
8199,01/12/2022,45.26
8200,01/13/2022,58.23
8201,01/14/2022,18.52
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `id`
- `date`
- `week_day`
- `week_number`

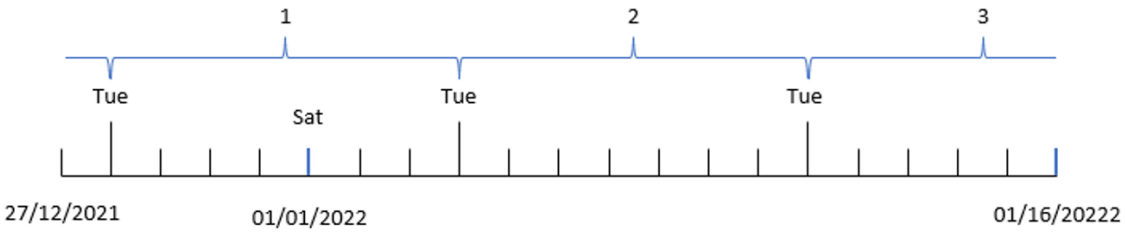
Ergebnistabelle

id	date	week_day	week_number
8183	12/27/2021	Mo	52
8184	12/28/2021	Di	1
8185	12/29/2021	Mi	1
8186	12/30/2021	Do	1
8187	12/31/2021	Fr	1
8188	01/01/2022	Sa	1
8189	01/02/2022	So	1
8190	01/03/2022	Mo	1
8191	01/04/2022	Di	2
8192	01/05/2022	Mi	2
8193	01/06/2022	Do	2
8194	01/07/2022	Fr	2
8195	01/08/2022	Sa	2
8196	01/09/2022	So	2

id	date	week_day	week_number
8197	01/10/2022	Mo	2
8198	01/11/2022	Di	3
8199	01/12/2022	Mi	3
8200	01/13/2022	Do	3
8201	01/14/2022	Fr	3

Der Parameter `first_week_date` wird auf 1 festgelegt, wodurch Dienstag zum ersten Tag der Woche wird. Der Parameter `broken_weeks` wird auf 0 festgelegt, was erzwingt, dass die Funktion nicht gestückelte Wochen verwendet. Im dritten Parameter wird abschließend der Parameter `reference_day` auf 2 festgelegt.

Diagramm der Funktion week(), Beispiel „reference_day“



Da die Funktion nicht gestückelte Wochen verwendet und ein Wert für `reference_day` von 2 als Parameter verwendet wird, muss Woche 1 nur zwei Tage im Januar enthalten. Da der erste Wochentag ein Dienstag ist, beginnt Woche 1 am 28. Dezember 2021 und endet am Montag, dem 3. Januar 2022.

Beispiel 5 – Diagrammobjektbeispiel

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript verwendet den gleichen Datensatz und das gleiche Szenario wie das erste Beispiel.

In diesem Beispiel wird jedoch der unveränderte Datensatz in die Anwendung geladen. Die Berechnung, die die Wochennummer zurückgibt, wird als Kennzahl in einem Diagrammobjekt erstellt.

Ladeskript

```
Transactions:
Load
*
Inline
[
id,date,amount
```

```
8183,12/27/2022,58.27
8184,12/28/2022,67.42
8185,12/29/2022,23.80
8186,12/30/2022,82.06
8187,12/31/2021,40.56
8188,01/01/2022,37.23
8189,01/02/2022,17.17
8190,01/03/2022,88.27
8191,01/04/2022,57.42
8192,01/05/2022,53.80
8193,01/06/2022,82.06
8194,01/07/2022,40.56
8195,01/08/2022,53.67
8196,01/09/2022,26.63
8197,01/10/2022,72.48
8198,01/11/2022,18.37
8199,01/12/2022,45.26
8200,01/13/2022,58.23
8201,01/14/2022,18.52
];
```

Ergebnisse

Gehen Sie folgendermaßen vor:

1. Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle.
2. Fügen Sie die folgenden Felder als Dimensionen hinzu:
   - `id`
   - `date`
3. Erstellen Sie dann die folgende Kennzahl:
   `=week (date)`
4. Erstellen Sie eine Kennzahl , `week_day`, um den Wochentagwert für jedes Transaktionsdatum anzuzeigen:
   `=weekday(date)`

Ergebnistabelle

id	date	=week(date)	=weekday(date)
8183	12/27/2021	53	Mo
8184	12/28/2021	53	Di
8185	12/29/2021	53	Mi
8186	12/30/2021	53	Do
8187	12/31/2021	53	Fr
8188	01/01/2022	1	Sa
8189	01/02/2022	2	So

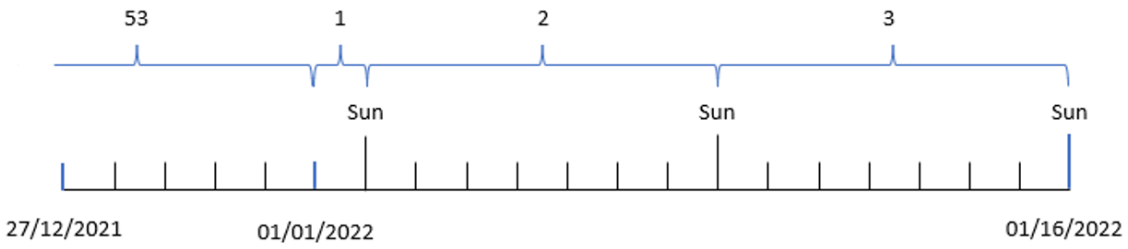
id	date	=week(date)	=weekday(date)
8190	01/03/2022	2	Mo
8191	01/04/2022	2	Di
8192	01/05/2022	2	Mi
8193	01/06/2022	2	Do
8194	01/07/2022	2	Fr
8195	01/08/2022	2	Sa
8196	01/09/2022	3	So
8197	01/10/2022	3	Mo
8198	01/11/2022	3	Di
8199	01/12/2022	3	Mi
8200	01/13/2022	3	Do
8201	01/14/2022	3	Fr

Das Feld `week_number` wird in der vorangehenden load-Anweisung erstellt, indem die Funktion `week()` verwendet und das Feld `date` als Argument der Funktion übergeben wird.

Es werden keine anderen Parameter an die Funktion übergeben, und daher gelten die folgenden, die Funktion `week()` betreffenden Standardvariablen:

- `BrokenWeeks`: Die Wochenzählung beginnt am 1. Januar.
- `FirstWeekDay`: Der erste Tag der Woche ist Sonntag.

Diagramm der Funktion week(), Diagrammobjektbeispiel



Da die Anwendung die Standardsystemvariable `BrokenWeeks` verwendet, beginnt Woche 1 am 1. Januar, einem Samstag.

Aufgrund der Standardsystemvariablen `FirstWeekDay` beginnen die Wochen an einem Sonntag. Der erste Sonntag nach dem 1. Januar ist der 2. Januar. Also beginnt an diesem Tag Woche 2.

Beispiel 6 – Szenario

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Datensatz mit einer Reihe von Transaktionen für die letzte Woche in 2019 und die ersten beiden Wochen in 2020, der in eine Tabelle namens `Transactions` geladen wird
- Datumsfeld, das im Format der Systemvariablen `DateFormat` (MM/TT/JJJJ) bereitgestellt wird

Die Anwendung verwendet in ihrem Dashboard hauptsächlich gestückelte Wochen. Der Endbenutzer möchte aber ein Diagrammobjekt, das den Gesamtumsatz nach Woche für nicht gestückelte Wochen darstellt. Der Referenztag soll der 2. Januar sein, und die Wochen sollen an einem Dienstag beginnen. Dies ist selbst dann möglich, wenn diese Dimension nicht im Datenmodell verfügbar ist. In diesem Fall wird die Funktion `week()` als berechnete Dimension im Diagramm verwendet.

Ladeskript

```
SET BrokenWeeks=1;
SET ReferenceDay=0;
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8183,12/27/2019,58.27
8184,12/28/2019,67.42
8185,12/29/2019,23.80
8186,12/30/2019,82.06
8187,12/31/2019,40.56
8188,01/01/2020,37.23
8189,01/02/2020,17.17
8190,01/03/2020,88.27
8191,01/04/2020,57.42
8192,01/05/2020,53.80
8193,01/06/2020,82.06
8194,01/07/2020,40.56
8195,01/08/2020,53.67
8196,01/09/2020,26.63
8197,01/10/2020,72.48
8198,01/11/2020,18.37
8199,01/12/2020,45.26
8200,01/13/2020,58.23
8201,01/14/2020,18.52
];
```

Ergebnisse

Gehen Sie folgendermaßen vor:

1. Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle.
2. Erstellen Sie die folgende berechnete Dimension:
   `=week(date)`
3. Erstellen Sie dann die folgende Aggregierungskennzahl:
   `=sum(amount)`
4. Legen Sie das **Zahlenformat** der Kennzahl auf **Währung** fest.
5. Wählen Sie das Menü **Sortieren** aus und entfernen Sie die benutzerdefinierte Sortierung für die berechnete Dimension.
6. Heben Sie die Auswahl der Optionen **Numerisch sortieren** und **Alphabetisch sortieren** auf.

Ergebnistabelle

week(date)	sum(amount)
52	$125.69
53	$146.42
1	$200.09
2	$347.57
3	$122.01

weekday

Diese Funktion liefert einen dualen Wert mit:

- Einem Wochentag wie in der Umgebungsvariable **DayNames** definiert.
- Einer ganzen Zahl zwischen 0-6 entsprechend den Tagen der Woche (0-6).

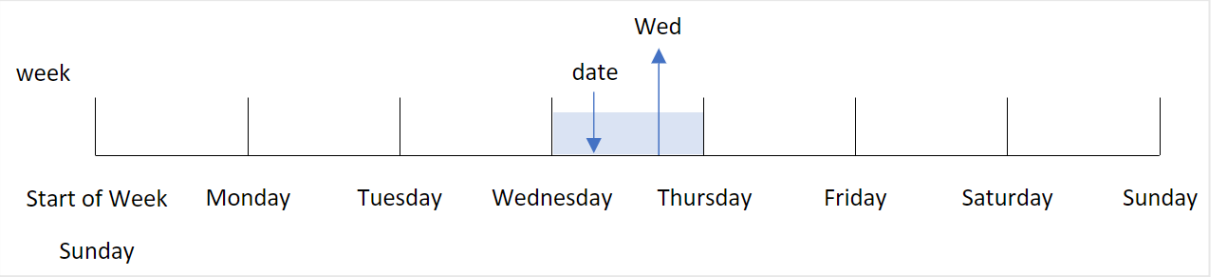
Syntax:

```
weekday(date [,first_week_day=0])
```

Rückgabe Datentyp: dual

Die Funktion `weekday()` bestimmt, auf welchen Tag der Woche ein Datum fällt. Dann gibt sie einen Stringwert zurück, der diesen Tag darstellt.

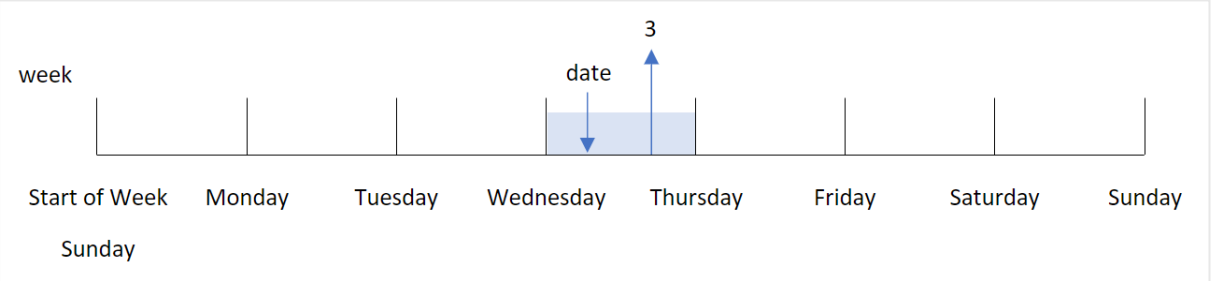
Diagramm der Funktion weekday(), die den Namen des Tages zurückgibt, auf den ein Datum fällt



Das Ergebnis gibt den Zahlenwert zurück, der dem betreffenden Tag der Woche (0-6) entspricht, basierend auf dem Starttag der Woche. Wenn beispielsweise der erste Tag der Woche auf Sonntag festgelegt ist, gibt ein Mittwoch einen Zahlenwert von `3` zurück. Dieser Starttag wird entweder durch die Systemvariable `FirstWeekDay` oder durch den Funktionsparameter `first_week_day` bestimmt.

Sie können diesen Zahlenwert als Teil einer arithmetischen Formel verwenden. Multiplizieren Sie sie beispielsweise mit `1`, um den Wert selbst zurückzugeben.

Diagramm der Funktion weekday(), in der der Zahlenwert des Tages anstelle des Namens angezeigt wird



Verwendung

Die Funktion `weekday()` ist nützlich, wenn Sie Aggregierungen nach Wochentag vergleichen möchten. Beispiel: Sie möchten den durchschnittlichen Umsatz nach Wochentag vergleichen.

Diese Dimensionen können im Ladeskript erstellt werden, indem die Funktion zum Erstellen eines Felds in einer **Master-Kalender**-Tabelle verwendet wird, oder sie können direkt in einem Diagramm als berechnete Kennzahl erstellt werden.

Verwandte Themen

Themen	Interaktion
FirstWeekDay (page 233)	Definiert den Starttag jeder Woche.

Argumente

Argument	Beschreibung
date	Datum oder Zeitstempel für die Evaluierung.
first_week_day	Legt den Tag fest, an dem die Woche beginnt. Ist nichts definiert, wird der Wert der Variable **FirstWeekDay** verwendet. *FirstWeekDay (page 233)*

Sie können die folgenden Werte verwenden, um den Tag des Wochenbeginns im Argument `first_week_day` festzulegen:

Werte für first_week_day

Tag	Wert
Montag	0
Dienstag	1
Mittwoch	2
Donnerstag	3
Freitag	4
Samstag	5
Sonntag	6

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Sofern nicht anders angegeben, ist `FirstWeekDay` in diesen Beispielen auf 0 eingestellt.

Funktionsbeispiele

Beispiel	Ergebnis
`weekday('10/12/1971')`	Gibt „Tue“ und 1 zurück.
`weekday('10/12/1971' , 6)`	Gibt „Tue“ und 2 zurück. In diesem Beispiel ist Sonntag (6) der erste Wochentag.

`SET FirstWeekDay=6;`
`...`
 	
`weekday('10/12/1971')`
	Gibt „Tue“ und 2 zurück.

Beispiel 1 – Wochentag-String

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz enthält eine Reihe von Transaktionen für 2022 und wird in eine Tabelle namens „`Transactions`“ geladen.
- Systemvariable `FirstWeekDay`, die ist auf 6 (Sonntag) festgelegt ist
- Variable `DayNames`, die auf die Verwendung der Standardtagesnamen festgelegt ist
- Ein vorangehender load-Befehl, der die Funktion `weekday()` enthält, die als das Feld „`week_day`“ festgelegt ist und den Wochentag zurückgibt, an dem die Transaktionen stattfanden

Ladeskript

```
SET DateFormat='MM/DD/YYYY';
SET DayNames='Mon;Tue;Wed;Thu;Fri;Sat;Sun';
SET FirstWeekDay=6;

Transactions:
    Load
       *,
       WeekDay(date) as week_day
       ;
Load
*
Inline
[
id,date,amount
8188,01/01/2022,37.23
8189,01/02/2022,17.17
8190,01/03/2022,88.27
8191,01/04/2022,57.42
8192,01/05/2022,53.80
```

```
8193,01/06/2022,82.06
8194,01/07/2022,40.39
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `id`
- `date`
- `week_day`

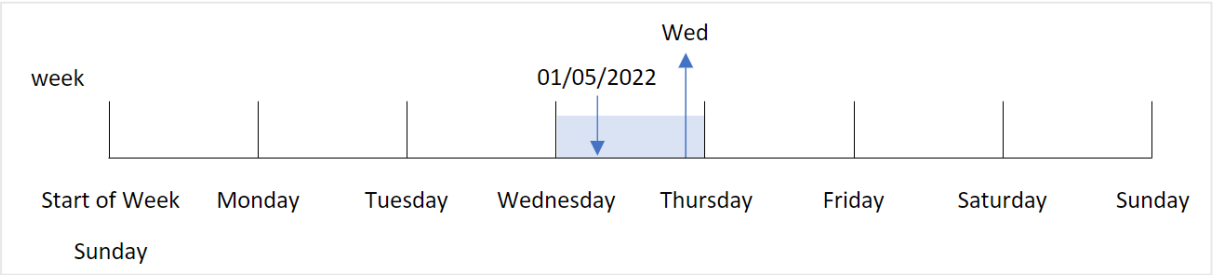
Ergebnistabelle

id	date	week_day
8188	01/01/2022	Sa
8189	01/02/2022	So
8190	01/03/2022	Mo
8191	01/04/2022	Di
8192	01/05/2022	Mi
8193	01/06/2022	Do
8194	01/07/2022	Fr

Das Feld „week_day“ wird in der vorangehenden load-Anweisung erstellt, indem die Funktion `weekday()` verwendet und das Datumsfeld als Argument der Funktion übergeben wird.

Die Funktion `weekday()` gibt den Wochentag-Stringwert zurück, also den Namen des Wochentages, der von der Systemvariablen `DayNames` festgelegt ist.

*Diagramm der Funktion **weekday()**, die Mittwoch als den Wochentag für Transaktion 8192 zurückgibt*



Transaktion 8192 fand am 5. Januar statt. Die Systemvariable `FirstWeekDay` legt den ersten Tag der Woche als Sonntag fest. Die Funktion `weekday()` bestimmt, dass die Transaktion an einem Mittwoch stattfand, und gibt diesen Wert in der abgekürzten Form der Systemvariablen `DayNames` im Feld `week_day` zurück.

Die Werte im Feld „week_day“ sind rechts in der Spalte ausgerichtet, da ein duales Zahlen- und Textergebnis für das Feld vorhanden ist (Mittwoch, 3). Um den Feldwert in den entsprechenden Zahlenwert zu konvertieren, kann das Feld in die Funktion num() eingeschlossen werden. Beispielsweise würde in Transaktion 8192 der Wert „Mittwoch“ in die Zahl 3 konvertiert.

Beispiel 2 – first_week_day

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Datensatz mit einer Reihe von Transaktionen für 2022, der in eine Tabelle namens Transactions geladen wird
- Systemvariable FirstWeekDay, die ist auf 6 (Sonntag) festgelegt ist
- Variable DayNames, die auf die Verwendung der Standardtagesnamen festgelegt ist
- Eine vorangehende load-Anweisung, die die Funktion weekday() enthält, die als das Feld „week_day“ festgelegt ist und den Wochentag zurückgibt, an dem die Transaktionen stattfanden

Ladeskript

```
SET DateFormat='MM/DD/YYYY';
SET DayNames='Mon;Tue;Wed;Thu;Fri;Sat;Sun';
SET FirstWeekDay=6;

Transactions:
    Load
       *,
       WeekDay(date,1) as week_day
       ;
Load
*
Inline
[
id,date,amount
8188,01/01/2022,37.23
8189,01/02/2022,17.17
8190,01/03/2022,88.27
8191,01/04/2022,57.42
8192,01/05/2022,53.80
8193,01/06/2022,82.06
8194,01/07/2022,40.39
];
```

Ergebnisse

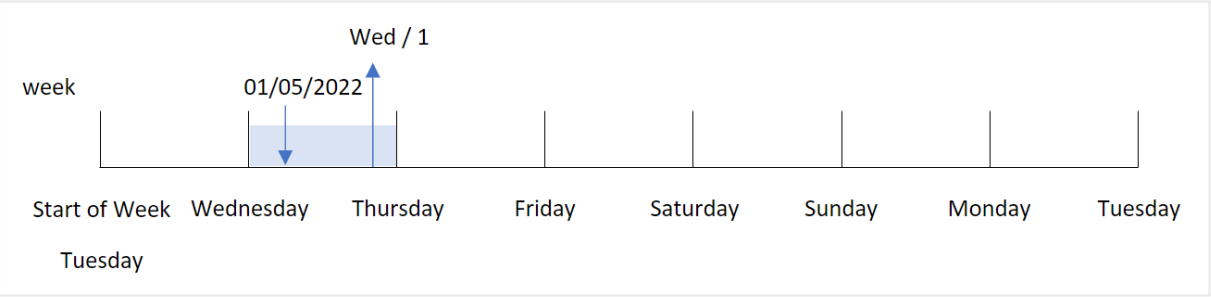
Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- id
- date
- week_day

Ergebnistabelle

id	date	week_day
8188	01/01/2022	Sa
8189	01/02/2022	So
8190	01/03/2022	Mo
8191	01/04/2022	Di
8192	01/05/2022	Mi
8193	01/06/2022	Do
8194	01/07/2022	Fr

Diagramm der Funktion weekday(), das zeigt, dass der Mittwoch den dualen Zahlenwert von 1 aufweist



Da das Argument first_week_day auf 1 in der Funktion weekday() festgelegt ist, ist Dienstag der erste Tag der Woche. Daher haben alle Transaktionen, die an einem Dienstag stattfinden, einen dualen Zahlenwert von 0.

Transaktion 8192 fand am 5. Januar statt. Die Funktion weekday() identifiziert, dass dies ein Mittwoch ist. Daher gibt die Formel den dualen Zahlenwert 1 zurück.

Beispiel 3 – Diagrammobjektbeispiel

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Datensatz mit einer Reihe von Transaktionen für 2022, der in eine Tabelle namens Transactions geladen wird

- Systemvariable `FirstWeekDay`, die auf 6 (Sonntag) festgelegt ist
- Variable `DayNames`, die auf die Verwendung der Standardtagesnamen festgelegt ist

In diesem Beispiel wird jedoch der unveränderte Datensatz in die Anwendung geladen. Die Berechnung, die den Wochentagwert identifiziert, wird als Kennzahl in einem Diagramm in der App erstellt.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';
SET DayNames='Mon;Tue;Wed;Thu;Fri;Sat;Sun';
SET FirstWeekDay=6;

Transactions:
Load
*
Inline
[
id,date,amount
8188,01/01/2022,37.23
8189,01/02/2022,17.17
8190,01/03/2022,88.27
8191,01/04/2022,57.42
8192,01/05/2022,53.80
8193,01/06/2022,82.06
8194,01/07/2022,40.39
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `id`
- `date`

Erstellen Sie die folgende Kennzahl, um den Wochentagwert zu berechnen:

- `=weekday(date)`

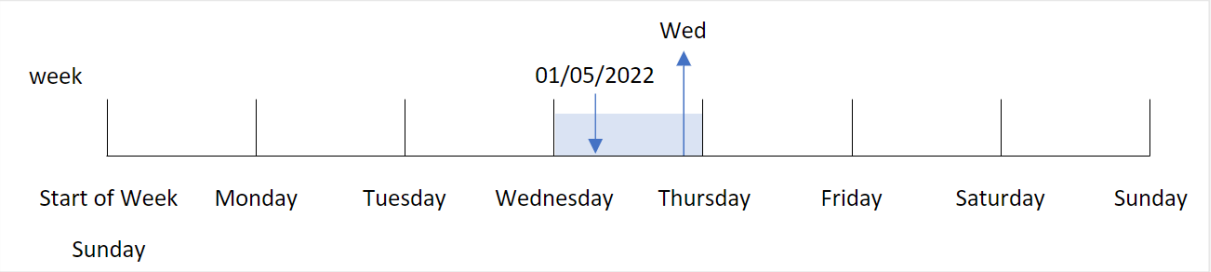
Ergebnistabelle

id	date	=weekday(date)
8188	01/01/2022	Sa
8189	01/02/2022	So
8190	01/03/2022	Mo
8191	01/04/2022	Di
8192	01/05/2022	Mi
8193	01/06/2022	Do
8194	01/07/2022	Fr

Das Feld „=weekday(date)“ wird im Diagramm erstellt, indem die Funktion weekday() verwendet und das Datumsfeld als Argument der Funktion übergeben wird.

Die Funktion weekday() gibt den Wochentag-Stringwert zurück, also den Namen des Wochentages, der von der Systemvariablen DayNames festgelegt ist.

Diagramm der Funktion weekday(), die Mittwoch als den Wochentag für Transaktion 8192 zurückgibt



Transaktion 8192 fand am 5. Januar statt. Die Systemvariable FirstWeekDay legt den ersten Tag der Woche als Sonntag fest. Die Funktion weekday() bestimmt, dass die Transaktion an einem Mittwoch stattfand, und gibt diesen Wert in der abgekürzten Form der Systemvariablen DayNames im Feld =weekday(date) zurück.

Beispiel 4 – Szenario

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Datensatz mit einer Reihe von Transaktionen für 2022, der in eine Tabelle namens Transactions geladen wird
- Systemvariable FirstWeekDay, die auf 6 (Sonntag) festgelegt ist
- Variable DayNames, die auf die Verwendung der Standardtagesnamen festgelegt ist

Der Endbenutzer möchte ein Diagramm, das den durchschnittlichen Umsatz nach Wochentag für die Transaktionen darstellt.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';
SET DayNames='Mon;Tue;Wed;Thu;Fri;Sat;Sun';
SET FirstWeekDay=6;

Transactions:
LOAD
  RecNo() AS id,
  MakeDate(2022, 1, Ceil(Rand() * 31)) as date,
  Rand() * 1000 AS amount

Autogenerate(1000);
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `=weekday(date)`
- `=avg(amount)`

Legen Sie das **Zahlenformat** der Kennzahl auf **Währung** fest.

Ergebnistabelle

weekday(date)	Avg(amount)
So	$536.96
Mo	$500.80
Di	$515.63
Mi	$509.21
Do	$482.70
Fr	$441.33
Sa	$505.22

weekend

Diese Funktion gibt einen Wert zurück, der dem Zeitstempel der letzten Millisekunde des letzten Tages der Kalenderwoche entspricht, in der **date** liegt. Das Ergebnis wird entsprechend dem im Skript definierten **DateFormat** formatiert.

Syntax:

```
WeekEnd(timestamp [, period_no [, first_week_day ]])
```

Rückgabe Datentyp: dual

Die Funktion `weekend()` bestimmt, in welche Woche das Datum fällt. Sie gibt dann einen Zeitstempel im Datumsformat für die letzte Millisekunde dieser Woche zurück. Der erste Tag der Woche wird durch die Umgebungsvariable `FirstWeekDay` bestimmt. Dies kann jedoch durch das Argument `first_week_day` in der Funktion `weekend()` überschrieben werden.

Argumente

Argument	Beschreibung
timestamp	Datum oder Zeitstempel für die Evaluierung.
period_no	**shift** ist eine ganze Zahl, wobei 0 für die Woche steht, die **date** enthält. Negative Werte von "shift" stehen für vorangehende Wochen, positive Werte für nachfolgende Wochen.

Argument	Beschreibung
first_week_day	Legt den Tag fest, an dem die Woche beginnt. Ist nichts definiert, wird der Wert der Variable **FirstWeekDay** verwendet. Die möglichen Werte für **first_week_day** sind 0 für Montag, 1 für Dienstag, 2 für Mittwoch, 3 für Donnerstag, 4 für Freitag, 5 für Samstag und 6 für Sonntag. Weitere Informationen über die Systemvariable finden Sie unter *FirstWeekDay (page 233)*.

Verwendung

Die Funktion `weekend()` wird in der Regel als Teil einer Formel verwendet, wenn in der Berechnung die verbleibenden Tage der Woche für das angegebene Datum verwendet werden sollen. Beispiel: Ein Benutzer möchte die gesamten, während der Woche noch nicht fällig gewordenen Zinsen berechnen.

Die folgenden Beispiele nehmen an:

```
SET FirstWeekDay=0;
```

Beispiel	Ergebnis
`weekend('01/10/2013')`	Gibt 01/12/2013 23:59:59 zurück.
`weekend('01/10/2013', -1)`	Gibt 01/05/2013 23:59:59. zurück.
`weekend('01/10/2013', 0, 1)`	Gibt 01/14/2013 23:59:59 zurück.

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Beispiele:

Wenn Sie ISO-Einstellungen für Wochen und Wochennummern verwenden möchten, müssen Sie Folgendes in das Skript einschließen:

```
Set DateFormat   ='YYYY-MM-DD';
Set FirstWeekDay =0;    // Monday as first week day
```

```
Set BrokenWeeks  =0;    //(use unbroken weeks)
Set ReferenceDay =4;    // Jan 4th is always in week 1
```

Wenn Sie US-Einstellungen verwenden möchten, müssen Sie Folgendes in das Skript einschließen:

```
Set DateFormat   ='M/D/YYYY';
Set FirstWeekDay =6;    // Sunday as first week day
Set BrokenWeeks  =1;    //(use broken weeks)
Set ReferenceDay =1;    // Jan 1st is always in week 1
```

Die obigen Beispiele führen zu folgenden Ergebnissen für die Funktion `weekend()`:

Beispiel der Funktion „Weekend“

Datum	ISO-Wochenende	US-Wochenende
Sat 2020 Dec 26	2020-12-27	12/26/2020
Sun 2020 Dec 27	2020-12-27	1/2/2021
Mon 2020 Dec 28	2021-01-03	1/2/2021
Tue 2020 Dec 29	2021-01-03	1/2/2021
Wed 2020 Dec 30	2021-01-03	1/2/2021
Thu 2020 Dec 31	2021-01-03	1/2/2021
Fri 2021 Jan 1	2021-01-03	1/2/2021
Sat 2021 Jan 2	2021-01-03	1/2/2021
Sun 2021 Jan 3	2021-01-03	1/9/2021
Mon 2021 Jan 4	2021-01-10	1/9/2021
Tue 2021 Jan 5	2021-01-10	1/9/2021

In der ISO-Spalte endet die Woche an Sonntagen, in der US-Spalte an Samstagen.

Beispiel 1 – einfaches Beispiel

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz enthält eine Reihe von Transaktionen für 2022 und wird in eine Tabelle namens `Transactions` geladen.
- Das Datumsfeld wird im Format der Systemvariablen `DateFormat` (MM/TT/JJJJ) bereitgestellt.
- Erstellung eines Felds `end_of_week`, das einen Zeitstempel für das Ende der Woche zurückgibt, in der die Transaktionen stattfanden

Ladeskript

```
SET FirstWeekDay=6;

Transactions:
    Load
        *,
        weekend(date) as end_of_week,
        timestamp(weekend(date)) as end_of_week_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `end_of_week`
- `end_of_week_timestamp`

Ergebnistabelle

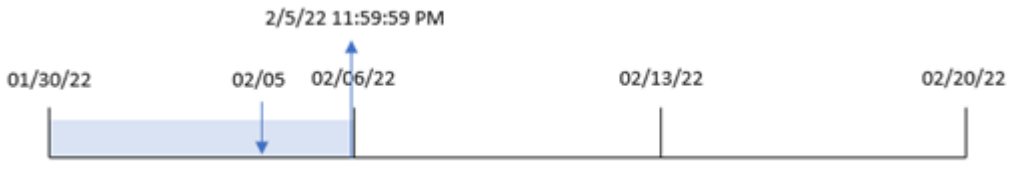
date	end_of_week	end_of_week_timestamp
1/7/2022	01/08/2022	1/8/2022 11:59:59 PM

date	end_of_week	end_of_week_timestamp
1/19/2022	01/22/2022	1/22/2022 11:59:59 PM
2/5/2022	02/05/2022	2/5/2022 11:59:59 PM
2/28/2022	03/05/2022	3/5/2022 11:59:59 PM
3/16/2022	03/19/2022	3/19/2022 11:59:59 PM
4/1/2022	04/02/2022	4/2/2022 11:59:59 PM
5/7/2022	05/07/2022	5/7/2022 11:59:59 PM
5/16/2022	05/21/2022	5/21/2022 11:59:59 PM
6/15/2022	06/18/2022	6/18/2022 11:59:59 PM
6/26/2022	07/02/2022	7/2/2022 11:59:59 PM
7/9/2022	07/09/2022	7/9/2022 11:59:59 PM
7/22/2022	07/23/2022	7/23/2022 11:59:59 PM
7/23/2022	07/23/2022	7/23/2022 11:59:59 PM
7/27/2022	07/30/2022	7/30/2022 11:59:59 PM
8/2/2022	08/06/2022	8/6/2022 11:59:59 PM
8/8/2022	08/13/2022	8/13/2022 11:59:59 PM
8/19/2022	08/20/2022	8/20/2022 11:59:59 PM
9/26/2022	10/01/2022	10/1/2022 11:59:59 PM
10/14/2022	10/15/2022	10/15/2022 11:59:59 PM
10/29/2022	10/29/2022	10/29/2022 11:59:59 PM

Das Feld `end_of_week` wird im vorangehenden load-Befehl erstellt, indem die Funktion `weekend()` verwendet und das Datumsfeld als Argument der Funktion übergeben wird.

Die Funktion `weekend()` identifiziert, in welche Woche der Datumswert fällt, und gibt einen Zeitstempel für die letzte Millisekunde dieser Woche zurück.

*Diagramm der Funktion **weekend()**, einfaches Beispiel*



Transaktion 8191 fand am 5. Februar statt. Die Systemvariable `FirstWeekDay` legt den ersten Tag der Woche als Sonntag fest. Die Funktion `weekend()` ermittelt, dass der erste Samstag nach dem 5. Februar – und somit nach dem Ende der Woche – der 5. Februar war. Daher gibt der Wert `end_of_week` für diese Transaktion die letzte Millisekunde dieses Tages zurück, also den 5. Februar um 11:59:59 PM.

Beispiel 2 – period_no

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Derselbe Datensatz und dasselbe Szenario wie im ersten Beispiel.
- Es wird ein Feld `previous_week_end` erstellt, das den Zeitstempel für den Start der Woche vor der Woche zurückgibt, in der die Transaktion stattfand.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        weekend(date,-1) as previous_week_end,
        timestamp(weekend(date,-1)) as previous_week_end_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

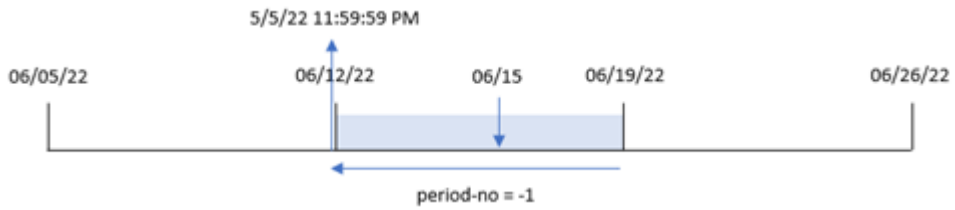
- `date`
- `previous_week_end`
- `previous_week_end_timestamp`

Ergebnistabelle

date	end_of_week	end_of_week_timestamp
1/7/2022	01/01/2022	1/1/2022 11:59:59 PM
1/19/2022	01/15/2022	1/15/2022 11:59:59 PM
2/5/2022	01/29/2022	1/29/2022 11:59:59 PM
2/28/2022	02/26/2022	2/26/2022 11:59:59 PM
3/16/2022	03/12/2022	3/12/2022 11:59:59 PM
4/1/2022	03/26/2022	3/26/2022 11:59:59 PM
5/7/2022	04/30/2022	4/30/2022 11:59:59 PM
5/16/2022	05/14/2022	5/14/2022 11:59:59 PM
6/15/2022	06/11/2022	6/11/2022 11:59:59 PM
6/26/2022	06/25/2022	6/25/2022 11:59:59 PM
7/9/2022	07/02/2022	7/2/2022 11:59:59 PM
7/22/2022	07/16/2022	7/16/2022 11:59:59 PM
7/23/2022	07/16/2022	7/16/2022 11:59:59 PM
7/27/2022	07/23/2022	7/23/2022 11:59:59 PM
8/2/2022	07/30/2022	7/30/2022 11:59:59 PM
8/8/2022	08/06/2022	8/6/2022 11:59:59 PM
8/19/2022	08/13/2022	8/13/2022 11:59:59 PM
9/26/2022	09/24/2022	9/24/2022 11:59:59 PM
10/14/2022	10/08/2022	10/8/2022 11:59:59 PM
10/29/2022	10/22/2022	10/22/2022 11:59:59 PM

Da in diesem Fall eine `period_no` von -1 als Versatzargument in der Funktion `weekend()` verwendet wurde, identifiziert die Funktion zuerst die Woche, in der die Transaktionen stattfinden. Dann geht sie eine Woche zurück und identifiziert die letzte Millisekunde dieser Woche.

Diagramm der Funktion weekend(), Beispiel „period_no“



Transaktion 8196 fand am 15. Juni statt. Die Funktion `weekend()` identifiziert, dass die Woche am 12. Juni beginnt. Daher endet die vorherige Woche am 11. Juni um 11:59:59 PM; dies ist der Wert, der für das Feld `previous_week_end` zurückgegeben wird.

Beispiel 3 – first_week_day

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript verwendet den gleichen Datensatz und das gleiche Szenario wie das erste Beispiel. In diesem Beispiel muss jedoch Dienstag als der erste Tag der Arbeitswoche festgelegt werden.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        weekend(date,0,1) as end_of_week,
        timestamp(weekend(date,0,1)) as end_of_week_timestamp,
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
```

```
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `end_of_week`
- `end_of_week_timestamp`

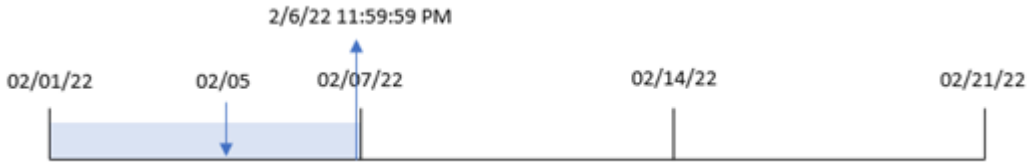
Ergebnistabelle

date	end_of_week	end_of_week_timestamp
1/7/2022	01/10/2022	1/10/2022 11:59:59 PM
1/19/2022	01/24/2022	1/24/2022 11:59:59 PM
2/5/2022	02/07/2022	2/7/2022 11:59:59 PM
2/28/2022	02/28/2022	2/28/2022 11:59:59 PM
3/16/2022	03/21/2022	3/21/2022 11:59:59 PM
4/1/2022	04/04/2022	4/4/2022 11:59:59 PM
5/7/2022	05/09/2022	5/9/2022 11:59:59 PM
5/16/2022	05/16/2022	5/16/2022 11:59:59 PM
6/15/2022	06/20/2022	6/20/2022 11:59:59 PM
6/26/2022	06/27/2022	6/27/2022 11:59:59 PM
7/9/2022	07/11/2022	7/11/2022 11:59:59 PM
7/22/2022	07/25/2022	7/25/2022 11:59:59 PM
7/23/2022	07/25/2022	7/25/2022 11:59:59 PM
7/27/2022	08/01/2022	8/1/2022 11:59:59 PM
8/2/2022	08/08/2022	8/8/2022 11:59:59 PM
8/8/2022	08/08/2022	8/8/2022 11:59:59 PM
8/19/2022	08/22/2022	8/22/2022 11:59:59 PM
9/26/2022	09/26/2022	9/26/2022 11:59:59 PM

date	end_of_week	end_of_week_timestamp
10/14/2022	10/17/2022	10/17/2022 11:59:59 PM
10/29/2022	10/31/2022	10/31/2022 11:59:59 PM

Da in diesem Fall das Argument `first_week_date` von 1 in der Funktion `weekend()` verwendet wird, legt sie Dienstag als ersten Tag der Woche fest.

Diagramm der Funktion weekend(), Beispiel „first_week_day“



Transaktion 8191 fand am 5. Februar statt. Die Funktion `weekend()` identifiziert, dass der erste Montag nach diesem Datum – und somit das Ende der Woche und der zurückgegebene Wert – der 6. Februar um 11:59:59 PM war.

Beispiel 4 – Diagrammobjektbeispiel

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript verwendet den gleichen Datensatz und das gleiche Szenario wie das erste Beispiel. In diesem Beispiel wird jedoch der unveränderte Datensatz in die Anwendung geladen. Die Berechnung, die einen Zeitstempel für das Ende der Woche zurückgibt, in dem die Transaktionen stattfanden, wird als Kennzahl in einem Diagrammobjekt der Anwendung erstellt.

Ladeskript

```
Transactions:
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
```

```
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu: `date`.

Um den Start der Woche zu berechnen, in der eine Transaktion stattfindet, erstellen Sie die folgenden Kennzahlen:

- `=weekend(date)`
- `=timestamp(weekend(date))`

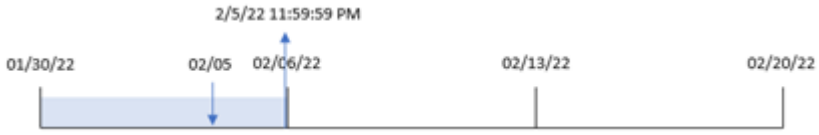
Ergebnistabelle

date	=weekend(date)	=timestamp(weekend(date))
1/7/2022	01/08/2022	1/8/2022 11:59:59 PM
1/19/2022	01/22/2022	1/22/2022 11:59:59 PM
2/5/2022	02/05/2022	2/5/2022 11:59:59 PM
2/28/2022	03/05/2022	3/5/2022 11:59:59 PM
3/16/2022	03/19/2022	3/19/2022 11:59:59 PM
4/1/2022	04/02/2022	4/2/2022 11:59:59 PM
5/7/2022	05/07/2022	5/7/2022 11:59:59 PM
5/16/2022	05/21/2022	5/21/2022 11:59:59 PM
6/15/2022	06/18/2022	6/18/2022 11:59:59 PM
6/26/2022	07/02/2022	7/2/2022 11:59:59 PM
7/9/2022	07/09/2022	7/9/2022 11:59:59 PM
7/22/2022	07/23/2022	7/23/2022 11:59:59 PM
7/23/2022	07/23/2022	7/23/2022 11:59:59 PM
7/27/2022	07/30/2022	7/30/2022 11:59:59 PM
8/2/2022	08/06/2022	8/6/2022 11:59:59 PM

date	=weekend(date)	=timestamp(weekend(date))
8/8/2022	08/13/2022	8/13/2022 11:59:59 PM
8/19/2022	08/20/2022	8/20/2022 11:59:59 PM
9/26/2022	10/01/2022	10/1/2022 11:59:59 PM
10/14/2022	10/15/2022	10/15/2022 11:59:59 PM
10/29/2022	10/29/2022	10/29/2022 11:59:59 PM

Die Kennzahl `end_of_week` wird im Diagrammobjekt erstellt, indem die Funktion `weekend()` verwendet und das Datumsfeld als Argument der Funktion übergeben wird. Die Funktion `weekend()` identifiziert, in welche Woche der Datumswert fällt, und gibt einen Zeitstempel für die letzte Millisekunde dieser Woche zurück.

Diagramm der Funktion weekend(), Diagrammobjektbeispiel



Transaktion 8191 fand am 5. Februar statt. Die Systemvariable `FirstweekDay` legt den ersten Tag der Woche als Sonntag fest. Die Funktion `weekend()` identifiziert den ersten Samstag nach dem 5. Februar – und somit nach dem Ende der Woche – als den 5. Februar. Daher gibt der Wert `end_of_week` für diese Transaktion die letzte Millisekunde dieses Tages zurück, also den 5. Februar um 11:59:59 PM.

Beispiel 5 – Szenario

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Datensatz, der in eine Tabelle namens `Employee_Expenses` geladen wird
- Daten bestehend aus Mitarbeiter-IDs, Mitarbeiternamen und den durchschnittlichen täglichen Spesenanträge jedes Mitarbeiters

Der Endbenutzer möchte ein Diagrammobjekt, das nach Mitarbeiter-ID und Mitarbeiternamen die geschätzten Spesenanträge anzeigt, die für die restliche Woche noch anfallen.

Ladeskript

```
Employee_Expenses:
Load
*
Inline
[
```

```
employee_id,employee_name,avg_daily_claim
182,Mark, $15
183,Deryck, $12.5
184,Dexter, $12.5
185,Sydney,$27
186,Agatha,$18
];
```

Ergebnisse

Gehen Sie folgendermaßen vor:

1. Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:
   - `employee_id`
   - `employee_name`
2. Erstellen Sie dann eine Kennzahl, um die kumulierten Zinsen zu berechnen: `=(weekend(today(1))-today(1))*avg_daily_claim`
3. Legen Sie das **Zahlenformat** der Kennzahl auf **Währung** fest.

Ergebnistabelle

employee_id	employee_name	=(weekend(today(1))-today(1))*avg_daily_claim
182	Mark	$90.00
183	Deryck	$75.00
184	Dexter	$75.00
185	Sydney	$162.00
186	Agatha	$108.00

Da das aktuelle Datum als einziges Argument verwendet wird, gibt die Funktion `weekend()` das Enddatum der aktuellen Woche zurück. Dann zieht die Formel das aktuelle Datum vom Enddatum der Woche ab und gibt die Anzahl der in dieser Woche verbleibenden Tage zurück.

Dieser Wert wird dann mit den durchschnittlichen täglichen Spesenanträgen der einzelnen Mitarbeitern multipliziert, um den geschätzten Spesenbetrag pro Mitarbeiter für den verbleibenden Teil der Woche zu berechnen.

weekname

Diese Funktion liefert den Zeitstempel der ersten Millisekunde der Kalenderwoche, in der **date** liegt. Das Ergebnis wird als Kombination von Jahr und Wochennummer formatiert.

Syntax:

```
WeekName(date[, period_no [, first_week_day [, broken_weeks [, reference_
day]]]])
```

Die Funktion weekname() bestimmt, in welche Woche das Datum fällt, und gibt die Wochennummer und das Jahr dieser Woche zurück. Der erste Tag der Woche wird durch die Systemvariable FirstWeekDay bestimmt. Sie können aber auch den ersten Tag der Woche ändern, indem Sie das Argument first_week_day in der Funktion weekname() verwenden.

In Qlik Sense werden die regionalen Einstellungen bei der Erstellung der App abgerufen, und die entsprechenden Einstellungen werden im Skript als Umgebungsvariablen gespeichert.

Ein nordamerikanischer App-Entwickler erhält oft Set BrokenWeeks=1; im Skript, was gestückelten Wochen entspricht. Ein europäischer App-Entwickler erhält oft Set BrokenWeeks=0; im Skript, was nicht gestückelten Wochen entspricht.

Wenn Ihre Anwendung gestückelte Wochen verwendet, beginnt die Zählung der Wochennummer am 1. Januar und endet am Tag vor der Systemvariablen FirstWeekDay, unabhängig davon, wie viele Tage verstrichen sind.

Wenn Ihre Anwendung jedoch vollständige Wochen verwendet, kann Woche 1 im Vorjahr oder in den ersten Tagen im Januar beginnen. Das hängt davon ab, wie Sie die Systemvariablen ReferenceDay und FirstWeekDay verwenden.

Beispiel der Funktion „Weekname“

Datum	ISO-Wochenname	US-Wochenname
Sat 2020 Dec 26	2020/52	2020/52
Sun 2020 Dec 27	2020/52	2020/53
Mon 2020 Dec 28	2020/53	2020/53
Tue 2020 Dec 29	2020/53	2020/53
Wed 2020 Dec 30	2020/53	2020/53
Thu 2020 Dec 31	2020/53	2020/53
Fri 2021 Jan 1	2020/53	2021/01
Sat 2021 Jan 2	2020/53	2021/01
Sun 2021 Jan 3	2020/53	2021/02
Mon 2021 Jan 4	2021/01	2021/02
Tue 2021 Jan 5	2021/01	2021/02

Verwendung

Die Funktion weekname() ist nützlich, wenn Sie Aggregierungen nach Wochen vergleichen möchten.

Das ist beispielsweise der Fall, wenn Sie den Gesamtumsatz von Produkten nach Woche anzeigen möchten. Um die Einheitlichkeit mit der Umgebungsvariablen BrokenWeeks in der Anwendung zu wahren, verwenden Sie weekname() anstelle von lunarweekname(). Wenn die Anwendung vollständige Wochen verwendet, kann Woche 1 Tage vom Dezember des Vorjahres enthalten oder Tage im Januar des laufenden Jahres ausschließen. Wenn die Anwendung unvollständige Wochen verwendet, kann Woche 1 weniger als sieben Tage enthalten.

Rückgabe Datentyp: dual

Argumente

Argument	Beschreibung
timestamp	Datum oder Zeitstempel für die Evaluierung.
period_no	**shift** ist eine ganze Zahl, wobei 0 für die Woche steht, die **date** enthält. Negative Werte von "shift" stehen für vorangehende Wochen, positive Werte für nachfolgende Wochen.
first_week_day	Legt den Tag fest, an dem die Woche beginnt. Ist nichts definiert, wird der Wert der Variable **FirstWeekDay** verwendet. Die möglichen Werte für **first_week_day** sind 0 für Montag, 1 für Dienstag, 2 für Mittwoch, 3 für Donnerstag, 4 für Freitag, 5 für Samstag und 6 für Sonntag. Weitere Informationen über die Systemvariable finden Sie unter *FirstWeekDay (page 233)*.
broken_weeks	Wenn Sie **broken_weeks** nicht angeben, wird der Wert der Variablen **BrokenWeeks** verwendet, um festzulegen, ob die Wochen gestückelt sind oder nicht.
reference_day	Wenn Sie **reference_day** nicht angeben, wird der Wert der Variablen **ReferenceDay** verwendet, um festzulegen, welcher Tag im Januar als Referenztag für die Definition von Woche 1 konfiguriert wird. Standardmäßig verwenden Qlik Sense-Funktionen 4 als Referenztag. Das heißt, dass die Woche 1 den 4. Januar enthalten muss, oder anders ausgedrückt: Woche 1 muss immer mindestens 4 Tage im Januar enthalten.

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Die folgenden Beispiele nehmen an:

```
Set FirstWeekDay=0;
Set BrokenWeeks=0;
Set ReferenceDay=4;
```

Funktionsbeispiele

Beispiel	Ergebnis
weekname('01/12/2013')	Liefert 2013/02.
weekname('01/12/2013', -1)	Returns 2013/01.
weekname('01/12/2013', 0, 1)	Liefert 2013/02.

Beispiel 1 – Datum ohne zusätzliche Argumente

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz, der eine Reihe von Transaktionen für die letzte Woche in 2021 und die ersten beiden Wochen in 2022 enthält, wird in eine Tabelle namens „`Transactions`“ geladen.
- Die Systemvariable `DateFormat`, die auf das Format `MM/DD/YYYY` festgelegt ist.
- Die Systemvariable `BrokenWeeks`, die auf `1` festgelegt ist.
- Die Systemvariable `FirstWeekDay`, die auf `6` festgelegt ist.
- Ein vorangehender load-Befehl, der Folgendes enthält:
  - Die Funktion `weekday()`, die als das Feld „`week_number`“ festgelegt ist, das das Jahr und die Wochennummer zurückgibt, in denen die Transaktion stattfand.
  - Die Funktion `weekname()`, die als das Feld mit dem Namen „`week_day`“ festgelegt ist, um den Wochentagwert für jedes Transaktionsdatum anzuzeigen.

Ladeskript

```
SET BrokenWeeks=1;
SET DateFormat='MM/DD/YYYY';
SET FirstWeekDay=6;

Transactions:
   Load
     *,
     WeekDay(date) as week_day,
     Weekname(date) as week_number
     ;
Load
*
Inline
[
id,date,amount
8183,12/27/2021,58.27
8184,12/28/2021,67.42
8185,12/29/2021,23.80
```

```
8186,12/30/2021,82.06
8187,12/31/2021,40.56
8188,01/01/2022,37.23
8189,01/02/2022,17.17
8190,01/03/2022,88.27
8191,01/04/2022,57.42
8192,01/05/2022,53.80
8193,01/06/2022,82.06
8194,01/07/2022,40.56
8195,01/08/2022,53.67
8196,01/09/2022,26.63
8197,01/10/2022,72.48
8198,01/11/2022,18.37
8199,01/12/2022,45.26
8200,01/13/2022,58.23
8201,01/14/2022,18.52
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `id`
- `date`
- `week_day`
- `week_number`

Ergebnistabelle

id	date	week_day	week_number
8183	12/27/2021	Mo	2021/53
8184	12/28/2021	Di	2021/53
8185	12/29/2021	Mi	2021/53
8186	12/30/2021	Do	2021/53
8187	12/31/2021	Fr	2021/53
8188	01/01/2022	Sa	2022/01
8189	01/02/2022	So	2022/02
8190	01/03/2022	Mo	2022/02
8191	01/04/2022	Di	2022/02
8192	01/05/2022	Mi	2022/02
8193	01/06/2022	Do	2022/02
8194	01/07/2022	Fr	2022/02

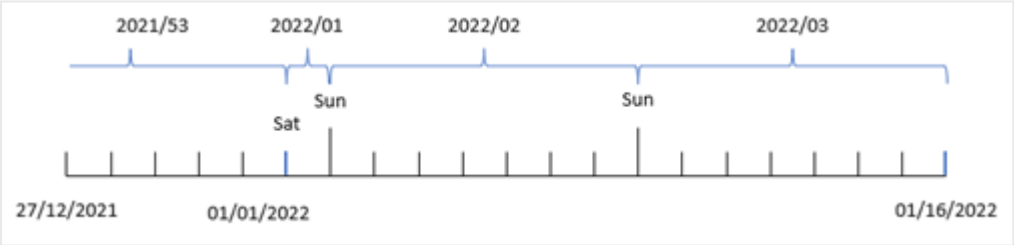
id	date	week_day	week_number
8195	01/08/2022	Sa	2022/02
8196	01/09/2022	So	2022/03
8197	01/10/2022	Mo	2022/03
8198	01/11/2022	Di	2022/03
8199	01/12/2022	Mi	2022/03
8200	01/13/2022	Do	2022/03
8201	01/14/2022	Fr	2022/03

Das Feld „week_number“ wird in der vorangehenden load-Anweisung erstellt, indem die Funktion weekname() verwendet und das Datumsfeld als Argument der Funktion übergeben wird.

Die Funktion weekname() identifiziert zunächst, in welche Woche der Datumswert fällt, und gibt die Wochennummernzählung und das Jahr zurück, in denen die Transaktion stattfand.

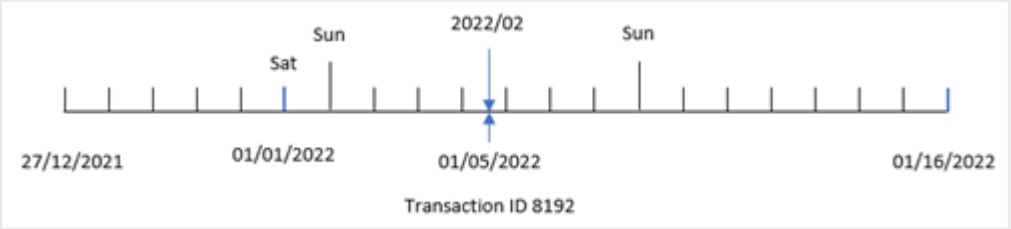
Die Systemvariable FirstWeekDay legt Sonntag als den ersten Tag der Woche fest. Die Systemvariable BrokenWeeks legt für die Anwendung die Verwendung von unvollständigen Wochen fest, Woche 1 beginnt also am 1. Januar.

Diagramm der Funktion weekname() mit den Standardvariablen.



Woche 1 beginnt am 1. Januar. Das ist ein Samstag, und daher geben Transaktionen an diesem Datum den Wert 2022/01 zurück (das Jahr und die Wochennummer).

Diagramm mit der Funktion weekname(), die die Wochennummer von Transaktion 8192 identifiziert.



Da die Anwendung unvollständige Wochen verwendet und der erste Wochentag Sonntag ist, geben Transaktionen vom 2. bis zum 8. Januar den Wert 2022/02 (Woche 2 im Jahr 2022) zurück. Ein Beispiel dafür ist Transaktion 8192, die am 5. Januar stattfand und den Wert 2022/02 für das Feld „week_number“ zurückgibt.

Beispiel 2 – period_no

Ladeskript und Ergebnisse

Übersicht

Es werden derselbe Datensatz und dasselbe Szenario wie im ersten Beispiel verwendet.

In diesem Beispiel besteht aber die Aufgabe darin, ein Feld „previous_week_number“ zu erstellen, das das Jahr und die Wochennummer vor Stattfinden der Transaktion zurückgibt.

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript in eine neue Registerkarte ein.

Ladeskript

```
SET BrokenWeeks=1;
SET FirstWeekDay=6;

Transactions:
   Load
     *,
     weekname(date,-1) as previous_week_number
     ;
Load
*
Inline
[
id,date,amount
8183,12/27/2021,58.27
8184,12/28/2021,67.42
8185,12/29/2021,23.80
8186,12/30/2021,82.06
8187,12/31/2021,40.56
8188,01/01/2022,37.23
8189,01/02/2022,17.17
8190,01/03/2022,88.27
8191,01/04/2022,57.42
8192,01/05/2022,53.80
8193,01/06/2022,82.06
8194,01/07/2022,40.56
8195,01/08/2022,53.67
8196,01/09/2022,26.63
8197,01/10/2022,72.48
8198,01/11/2022,18.37
8199,01/12/2022,45.26
8200,01/13/2022,58.23
8201,01/14/2022,18.52
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

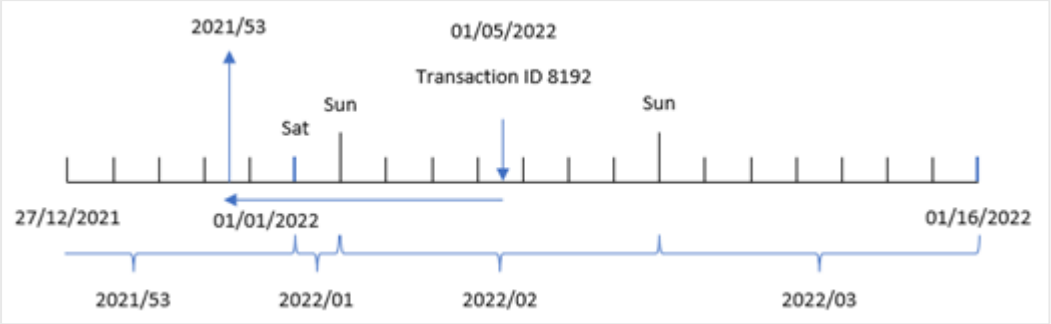
- `id`
- `date`
- `week_day`
- `week_number`

Ergebnistabelle

id	date	week_day	week_number
8183	12/27/2021	Mo	2021/52
8184	12/28/2021	Di	2021/52
8185	12/29/2021	Mi	2021/52
8186	12/30/2021	Do	2021/52
8187	12/31/2021	Fr	2021/52
8188	01/01/2022	Sa	2021/52
8189	01/02/2022	So	2021/53
8190	01/03/2022	Mo	2021/53
8191	01/04/2022	Di	2021/53
8192	01/05/2022	Mi	2021/53
8193	01/06/2022	Do	2021/53
8194	01/07/2022	Fr	2021/53
8195	01/08/2022	Sa	2022/01
8196	01/09/2022	So	2022/02
8197	01/10/2022	Mo	2022/02
8198	01/11/2022	Di	2022/02
8199	01/12/2022	Mi	2022/02
8200	01/13/2022	Do	2022/02
8201	01/14/2022	Fr	2022/02

Da eine `period_no` von `-1` als Versatzargument in der Funktion `weekname()` verwendet wurde, identifiziert die Funktion zuerst die Woche, in der die Transaktionen stattfanden. Dann geht sie eine Woche zurück und identifiziert die erste Millisekunde dieser Woche.

Diagramm der Funktion weekname() mit einem Versatz für period_no von -1.



Transaktion 8192 fand am 5. Januar 2022 statt. Die Funktion weekname() sucht eine Woche vorher, 30. Dezember 2021, und gibt die Wochennummer und das Jahr für dieses Datum zurück – 2021/53.

Beispiel 3 – first_week_day

Ladeskript und Ergebnisse

Übersicht

Es werden derselbe Datensatz und dasselbe Szenario wie im ersten Beispiel verwendet.

In diesem Beispiel soll aber laut der Unternehmensrichtlinie die Arbeitswoche am Dienstag beginnen.

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript in eine neue Registerkarte ein.

Ladeskript

```
SET BrokenWeeks=1;
SET DateFormat='MM/DD/YYYY';

Transactions:
   Load
     *,
     weekday(date) as week_day,
     weekname(date,0,1) as week_number
     ;
Load
*
Inline
[
id,date,amount
8183,12/27/2021,58.27
8184,12/28/2021,67.42
8185,12/29/2021,23.80
8186,12/30/2021,82.06
8187,12/31/2021,40.56
8188,01/01/2022,37.23
8189,01/02/2022,17.17
8190,01/03/2022,88.27
8191,01/04/2022,57.42
8192,01/05/2022,53.80
```

```
8193,01/06/2022,82.06
8194,01/07/2022,40.56
8195,01/08/2022,53.67
8196,01/09/2022,26.63
8197,01/10/2022,72.48
8198,01/11/2022,18.37
8199,01/12/2022,45.26
8200,01/13/2022,58.23
8201,01/14/2022,18.52
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

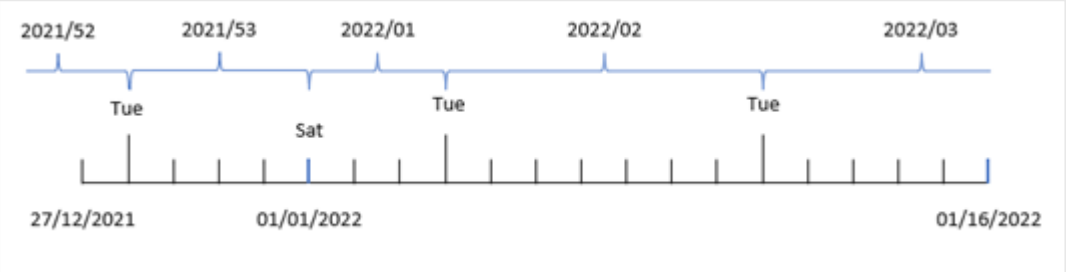
- `id`
- `date`
- `week_day`
- `week_number`

Ergebnistabelle

id	date	week_day	week_number
8183	12/27/2021	Mo	2021/52
8184	12/28/2021	Di	2021/53
8185	12/29/2021	Mi	2021/53
8186	12/30/2021	Do	2021/53
8187	12/31/2021	Fr	2021/53
8188	01/01/2022	Sa	2022/01
8189	01/02/2022	So	2022/01
8190	01/03/2022	Mo	2022/01
8191	01/04/2022	Di	2022/02
8192	01/05/2022	Mi	2022/02
8193	01/06/2022	Do	2022/02
8194	01/07/2022	Fr	2022/02
8195	01/08/2022	Sa	2022/02
8196	01/09/2022	So	2022/02
8197	01/10/2022	Mo	2022/02
8198	01/11/2022	Di	2022/03

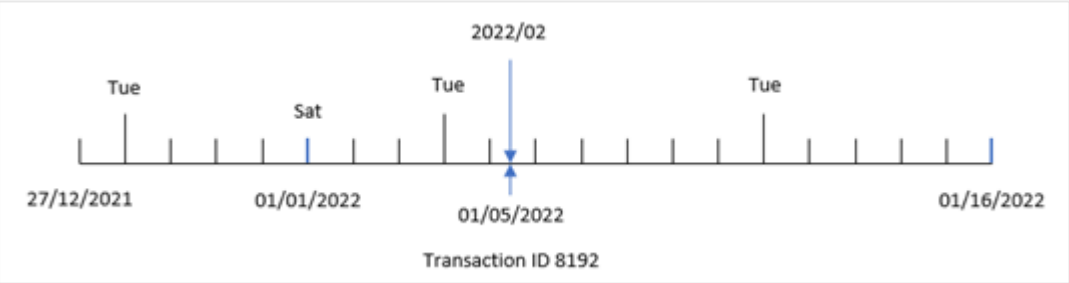
id	date	week_day	week_number
8199	01/12/2022	Mi	2022/03
8200	01/13/2022	Do	2022/03
8201	01/14/2022	Fr	2022/03

Diagramm der Funktion weekname() mit Dienstag als erstem Tag der Woche.



Da das Argument `first_week_date` von 1 in der Funktion `weekname()` verwendet wird, nutzt sie Dienstag als ersten Tag der Woche. Somit bestimmt die Funktion, dass Woche 53 im Jahr 2021 am Dienstag, den 28. Dezember beginnt, und da unvollständige Wochen verwendet werden, beginnt Woche 1 am 1. Januar 2022 und endet in der letzten Millisekunde von Montag, 3. Januar 2022.

Das Diagramm zeigt die Wochennummer von Transaktion 8192 mit Dienstag als erstem Tag der Woche.



Transaktion 8192 fand am 5. Januar 2022 statt. Wenn also ein Parameter `first_week_day` von Dienstag verwendet wird, gibt die Funktion `weekname()` den Wert `2022/02` für das Feld „`week_number`“ zurück.

Beispiel 4 – Diagrammobjektbeispiel

Ladeskript und Diagrammformel

Übersicht

Es werden derselbe Datensatz und dasselbe Szenario wie im ersten Beispiel verwendet.

In diesem Beispiel wird jedoch der unveränderte Datensatz in die Anwendung geladen. Die Berechnung, die die Jahreszahl für die Woche zurückgibt, in der die Transaktionen stattfanden, wird als Kennzahl in einem Diagrammobjekt der Anwendung erstellt.

Ladeskript

```
SET BrokenWeeks=1;
Transactions:
Load
*
Inline
[
id,date,amount
8183,12/27/2021,58.27
8184,12/28/2021,67.42
8185,12/29/2021,23.80
8186,12/30/2021,82.06
8187,12/31/2021,40.56
8188,01/01/2022,37.23
8189,01/02/2022,17.17
8190,01/03/2022,88.27
8191,01/04/2022,57.42
8192,01/05/2022,53.80
8193,01/06/2022,82.06
8194,01/07/2022,40.56
8195,01/08/2022,53.67
8196,01/09/2022,26.63
8197,01/10/2022,72.48
8198,01/11/2022,18.37
8199,01/12/2022,45.26
8200,01/13/2022,58.23
8201,01/14/2022,18.52
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `id`
- `date`
- `=week_day (date)`

Um den Start der Woche zu berechnen, in dem eine Transaktion stattfindet, erstellen Sie die folgende Kennzahl:

```
=weekname(date)
```

Ergebnistabelle

id	date	=weekday(date)	=weekname(date)
8183	12/27/2021	Mo	2021/53
8184	12/28/2021	Di	2021/53
8185	12/29/2021	Mi	2021/53

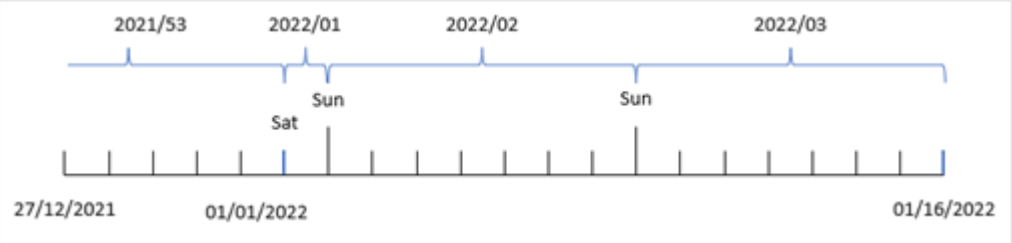
id	date	=weekday(date)	=weekname(date)
8186	12/30/2021	Do	2021/53
8187	12/31/2021	Fr	2021/53
8188	01/01/2022	Sa	2022/01
8189	01/02/2022	So	2022/02
8190	01/03/2022	Mo	2022/02
8191	01/04/2022	Di	2022/02
8192	01/05/2022	Mi	2022/02
8193	01/06/2022	Do	2022/02
8194	01/07/2022	Fr	2022/02
8195	01/08/2022	Sa	2022/02
8196	01/09/2022	So	2022/03
8197	01/10/2022	Mo	2022/03
8198	01/11/2022	Di	2022/03
8199	01/12/2022	Mi	2022/03
8200	01/13/2022	Do	2022/03
8201	01/14/2022	Fr	2022/03

Das Feld „week_number“ wird als Kennzahl im Diagrammobjekt erstellt, indem die Funktion `weekname()` verwendet und das Datumsfeld als Argument der Funktion übergeben wird.

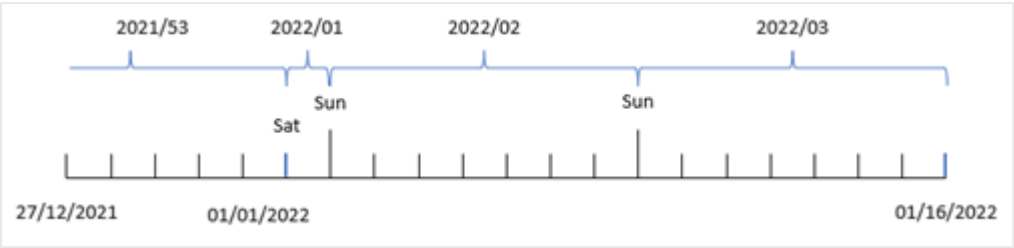
Die Funktion `weekname()` identifiziert zunächst, in welche Woche der Datumswert fällt, und gibt die Wochennummernzählung und das Jahr zurück, in denen die Transaktion stattfindet.

Die Systemvariable `FirstWeekDay` legt Sonntag als den ersten Tag der Woche fest. Die Systemvariable `BrokenWeeks` legt für die Anwendung die Verwendung von unvollständigen Wochen fest, Woche 1 beginnt also am 1. Januar.

Das Diagramm zeigt die Wochennummer mit Sonntag als erstem Tag der Woche.



Das Diagramm zeigt, dass die Transaktion 8192 in Woche 2 stattfand.



Da die Anwendung unvollständige Wochen verwendet und der erste Wochentag Sonntag ist, geben Transaktionen vom 2. bis zum 8. Januar den Wert „2022/02“ (Woche 2 im Jahr 2022) zurück. Sie sehen, dass die Transaktion 8192 am 5. Januar stattfand und den Wert „2022/02“ für das Feld „`week_number`“ zurückgibt.

Beispiel 5 – Szenario

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz, der eine Reihe von Transaktionen für die letzte Woche in 2019 und die ersten beiden Wochen in 2020 enthält, wird in eine Tabelle namens „`Transactions`“ geladen.
- Die Systemvariable `BrokenWeeks`, die auf `0` festgelegt ist.
- Die Systemvariable `ReferenceDay`, die auf `2` festgelegt ist.
- Die Systemvariable `DateFormat`, die auf das Format `MM/DD/YYYY` festgelegt ist.

Ladeskript

```
SET BrokenWeeks=0;
SET ReferenceDay=2;
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8183,12/27/2019,58.27
8184,12/28/2019,67.42
8185,12/29/2019,23.80
8186,12/30/2019,82.06
8187,12/31/2019,40.56
8188,01/01/2020,37.23
8189,01/02/2020,17.17
8190,01/03/2020,88.27
8191,01/04/2020,57.42
8192,01/05/2020,53.80
```

```
8193,01/06/2020,82.06
8194,01/07/2020,40.56
8195,01/08/2020,53.67
8196,01/09/2020,26.63
8197,01/10/2020,72.48
8198,01/11/2020,18.37
8199,01/12/2020,45.26
8200,01/13/2020,58.23
8201,01/14/2020,18.52
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle.

Erstellen Sie eine berechnete Dimension anhand der folgenden Formel:

```
=weekname(date)
```

Erstellen Sie die folgende Aggregierungskennzahl, um den Gesamtumsatz zu berechnen:

```
=sum(amount)
```

Legen Sie das **Zahlenformat** der Kennzahl auf **Währung** fest.

Ergebnistabelle

weekname(date)	=sum(amount)
2019/52	$125.69
2020/01	$346.51
2020/02	$347.57
2020/03	$122.01

Um die Ergebnisse der Verwendung der Funktion „weekname()“ in diesem Szenario zu zeigen, fügen Sie folgendes Feld als Dimension hinzu:

```
date
```

Ergebnistabelle mit Datumsfeld

weekname(date)	date	=sum(amount)
2019/52	12/27/2019	$58.27
2019/52	12/28/2019	$67.42
2020/01	12/29/2019	$23.80
2020/01	12/30/2019	$82.06
2020/01	12/31/2019	$40.56
2020/01	01/01/2020	$37.23

weekname(date)	date	=sum(amount)
2020/01	01/02/2020	$17.17
2020/01	01/03/2020	$88.27
2020/01	01/04/2020	$57.42
2020/02	01/05/2020	$53.80
2020/02	01/06/2020	$82.06
2020/02	01/07/2020	$40.56
2020/02	01/08/2020	$53.67
2020/02	01/09/2020	$26.63
2020/02	01/10/2020	$72.48
2020/02	01/11/2020	$18.37
2020/03	01/12/2020	$45.26
2020/03	01/13/2020	$58.23
2020/03	01/14/2020	$18.52

Da die Anwendung vollständige Wochen verwendet und Woche 1 aufgrund der Systemvariablen `ReferenceDay` mindestens zwei Tage im Januar benötigt, enthält Woche 1 für 2020 Transaktionen vom 29. Dezember 2019.

weekstart

Diese Funktion liefert den Zeitstempel der ersten Millisekunde des ersten Tags der Kalenderwoche, in der **date** liegt. Das Ergebnis wird entsprechend dem im Skript definierten **DateFormat** formatiert.

Syntax:

```
WeekStart(timestamp [, period_no [, first_week_day ]])
```

Rückgabe Datentyp: dual

Die Funktion `weekstart()` bestimmt, in welche Woche das Datum fällt. Sie gibt dann einen Zeitstempel im Datumsformat für die erste Millisekunde dieser Woche zurück. Der erste Tag der Woche wird durch die Umgebungsvariable `FirstWeekDay` bestimmt. Dies kann jedoch durch das Argument `first_week_day` in der Funktion `weekstart()` überschrieben werden.

Argumente

Argument	Beschreibung
timestamp	Datum oder Zeitstempel für die Evaluierung.
period_no	**shift** ist eine ganze Zahl, wobei 0 für die Woche steht, die **date** enthält. Negative Werte von "shift" stehen für vorangehende Wochen, positive Werte für nachfolgende Wochen.

Argument	Beschreibung
first_week_day	Legt den Tag fest, an dem die Woche beginnt. Ist nichts definiert, wird der Wert der Variable **FirstWeekDay** verwendet. Die möglichen Werte für **first_week_day** sind 0 für Montag, 1 für Dienstag, 2 für Mittwoch, 3 für Donnerstag, 4 für Freitag, 5 für Samstag und 6 für Sonntag. Weitere Informationen über die Systemvariable finden Sie unter *FirstWeekDay (page 233)*.

Verwendung

Die Funktion `weekstart()` wird in der Regel als Teil einer Formel verwendet, wenn in der Berechnung der Teil der Woche verwendet werden soll, der bereits verstrichen ist. Beispielsweise kann sie verwendet werden, um den Gesamtlohn zu berechnen, der bisher von den Mitarbeitern in der Woche verdient wurde.

Die folgenden Beispiele nehmen an:

```
SET FirstWeekDay=0;
```

Funktionsbeispiele

Beispiel	Ergebnis
`weekstart('01/12/2013')`	Gibt 01/07/2013 zurück.
`weekstart('01/12/2013', -1 )`	Gibt 11/31/2012 zurück.
`weekstart('01/12/2013', 0, 1)`	Gibt 01/08/2013 zurück.

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Beispiele:

Wenn Sie ISO-Einstellungen für Wochen und Wochennummern verwenden möchten, müssen Sie Folgendes in das Skript einschließen:

```
Set DateFormat   ='YYYY-MM-DD';
Set FirstWeekDay =0;    // Monday as first week day
```

```
Set BrokenWeeks  =0;    //(use unbroken weeks)
Set ReferenceDay =4;    // Jan 4th is always in week 1
```

Wenn Sie US-Einstellungen verwenden möchten, müssen Sie Folgendes in das Skript einschließen:

```
Set DateFormat   ='M/D/YYYY';
Set FirstWeekDay =6;    // Sunday as first week day
Set BrokenWeeks  =1;    //(use broken weeks)
Set ReferenceDay =1;    // Jan 1st is always in week 1
```

Die obigen Beispiele führen zu folgenden Ergebnissen für die Funktion `weekstart()`:

Beispiel der Funktion „Weekstart“

Datum	ISO-Wochenstart	US-Wochenstart
Sat 2020 Dec 26	2020-12-21	12/20/2020
Sun 2020 Dec 27	2020-12-21	12/27/2020
Mon 2020 Dec 28	2020-12-28	12/27/2020
Tue 2020 Dec 29	2020-12-28	12/27/2020
Wed 2020 Dec 30	2020-12-28	12/27/2020
Thu 2020 Dec 31	2020-12-28	12/27/2020
Fri 2021 Jan 1	2020-12-28	12/27/2020
Sat 2021 Jan 2	2020-12-28	12/27/2020
Sun 2021 Jan 3	2020-12-28	1/3/2021
Mon 2021 Jan 4	2021-01-04	1/3/2021
Tue 2021 Jan 5	2021-01-04	1/3/2021

In der ISO-Spalte beginnt die Woche an Montagen, in der US-Spalte an Sonntagen.

Beispiel 1 – keine zusätzlichen Argumente

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz enthält eine Reihe von Transaktionen für 2022 und wird in eine Tabelle namens `Transactions` geladen.
- Das Datumsfeld wird im Format der Systemvariablen `DateFormat` (MM/TT/JJJJ) bereitgestellt.
- Erstellung eines Felds `start_of_week`, das den Zeitstempel für den Start der Woche zurückgibt, in der die Transaktionen stattfanden

Ladeskript

```
SET FirstWeekDay=6;

Transactions:
    Load
        *,
        weekstart(date) as start_of_week,
        timestamp(weekstart(date)) as start_of_week_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `start_of_week`
- `start_of_week_timestamp`

Ergebnistabelle

date	start_of_week	start_of_week_timestamp
1/7/2022	01/02/2022	1/2/2022 12:00:00 AM

date	start_of_week	start_of_week_timestamp
1/19/2022	01/16/2022	1/16/2022 12:00:00 AM
2/5/2022	01/30/2022	1/30/2022 12:00:00 AM
2/28/2022	02/27/2022	2/27/2022 12:00:00 AM
3/16/2022	03/13/2022	3/13/2022 12:00:00 AM
4/1/2022	03/27/2022	3/27/2022 12:00:00 AM
5/7/2022	05/01/2022	5/1/2022 12:00:00 AM
5/16/2022	05/15/2022	5/15/2022 12:00:00 AM
6/15/2022	06/12/2022	6/12/2022 12:00:00 AM
6/26/2022	06/26/2022	6/26/2022 12:00:00 AM
7/9/2022	07/03/2022	7/3/2022 12:00:00 AM
7/22/2022	07/17/2022	7/17/2022 12:00:00 AM
7/23/2022	07/17/2022	7/17/2022 12:00:00 AM
7/27/2022	07/24/2022	7/24/2022 12:00:00 AM
8/2/2022	07/31/2022	7/31/2022 12:00:00 AM
8/8/2022	08/07/2022	8/7/2022 12:00:00 AM
8/19/2022	08/14/2022	8/14/2022 12:00:00 AM
9/26/2022	09/25/2022	9/25/2022 12:00:00 AM
10/14/2022	10/09/2022	10/9/2022 12:00:00 AM
10/29/2022	10/23/2022	10/23/2022 12:00:00 AM

Das Feld `start_of_week` wird im vorangehenden load-Befehl erstellt, indem die Funktion `weekstart()` verwendet und das Datumsfeld als Argument der Funktion übergeben wird.

Die Funktion `weekstart()` identifiziert zunächst, in welche Woche der Datumswert fällt, und gibt einen Zeitstempel für die erste Millisekunde dieser Woche zurück.

Diagramm der Funktion `weekstart()`, Beispiel ohne zusätzliche Argumente

Transaktion 8191 fand am 5. Februar statt. Die Systemvariable `FirstWeekDay` legt den ersten Tag der Woche als Sonntag fest. Die Funktion `weekstart()` identifiziert den ersten Sonntag vor dem 5. Februar – und somit dem Start der Woche – als den 30. Januar. Daher gibt der Wert `start_of_week` für diese Transaktion die erste Millisekunde dieses Tages zurück, also den 30. Januar um 12:00:00 AM.

Beispiel 2 – period_no

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Derselbe Datensatz und dasselbe Szenario wie im ersten Beispiel.
- Es wird ein Feld `previous_week_start` erstellt, das den Zeitstempel für den Start des Quartals vor dem Quartal zurückgibt, in dem die Transaktion stattfand.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        weekstart(date,-1) as previous_week_start,
        timestamp(weekstart(date,-1)) as previous_week_start_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

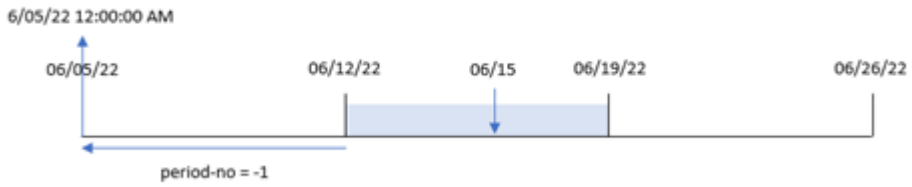
- `date`
- `previous_week_start`
- `previous_week_start_timestamp`

Ergebnistabelle

date	previous_week_start	previous_week_start_timestamp
1/7/2022	12/26/2021	12/26/2021 12:00:00 AM
1/19/2022	01/09/2022	1/9/2022 12:00:00 AM
2/5/2022	01/23/2022	1/23/2022 12:00:00 AM
2/28/2022	02/20/2022	2/20/2022 12:00:00 AM
3/16/2022	03/06/2022	3/6/2022 12:00:00 AM
4/1/2022	03/20/2022	3/20/2022 12:00:00 AM
5/7/2022	04/24/2022	4/24/2022 12:00:00 AM
5/16/2022	05/08/2022	5/8/2022 12:00:00 AM
6/15/2022	06/05/2022	6/5/2022 12:00:00 AM
6/26/2022	06/19/2022	6/19/2022 12:00:00 AM
7/9/2022	06/26/2022	6/26/2022 12:00:00 AM
7/22/2022	07/10/2022	7/10/2022 12:00:00 AM
7/23/2022	07/10/2022	7/10/2022 12:00:00 AM
7/27/2022	07/17/2022	7/17/2022 12:00:00 AM
8/2/2022	07/24/2022	7/24/2022 12:00:00 AM
8/8/2022	07/31/2022	7/31/2022 12:00:00 AM
8/19/2022	08/07/2022	8/7/2022 12:00:00 AM
9/26/2022	09/18/2022	9/18/2022 12:00:00 AM
10/14/2022	10/02/2022	10/2/2022 12:00:00 AM
10/29/2022	10/16/2022	10/16/2022 12:00:00 AM

Da in dieser Instanz eine `period_no` von -1 als Versatzargument in der Funktion `weekstart()` verwendet wurde, identifiziert die Funktion zuerst die Woche, in der die Transaktionen stattfanden. Dann geht sie eine Woche zurück und identifiziert die erste Millisekunde dieser Woche.

Diagramm der Funktion weekstart(), Beispiel „period_no“



Transaktion 8196 fand am 15. Juni statt. Die Funktion `weekstart()` identifiziert, dass die Woche am 12. Juni beginnt. Daher begann die vorherige Woche am 5. Juni um 12:00:00 AM; dies ist der Wert, der für das Feld `previous_week_start` zurückgegeben wird.

Beispiel 3 – first_week_day

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript verwendet den gleichen Datensatz und das gleiche Szenario wie das erste Beispiel. In diesem Beispiel muss jedoch Dienstag als der erste Tag der Arbeitswoche festgelegt werden.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        weekstart(date,0,1) as start_of_week,
        timestamp(weekstart(date,0,1)) as start_of_week_timestamp
        ;
Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
```

```
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `start_of_week`
- `start_of_week_timestamp`

Ergebnistabelle

date	start_of_week	start_of_week_timestamp
1/7/2022	01/04/2022	1/4/2022 12:00:00 AM
1/19/2022	01/18/2022	1/18/2022 12:00:00 AM
2/5/2022	02/01/2022	2/1/2022 12:00:00 AM
2/28/2022	02/22/2022	2/22/2022 12:00:00 AM
3/16/2022	03/15/2022	3/15/2022 12:00:00 AM
4/1/2022	03/29/2022	3/29/2022 12:00:00 AM
5/7/2022	05/03/2022	5/3/2022 12:00:00 AM
5/16/2022	05/10/2022	5/10/2022 12:00:00 AM
6/15/2022	06/14/2022	6/14/2022 12:00:00 AM
6/26/2022	06/21/2022	6/21/2022 12:00:00 AM
7/9/2022	07/05/2022	7/5/2022 12:00:00 AM
7/22/2022	07/19/2022	7/19/2022 12:00:00 AM
7/23/2022	07/19/2022	7/19/2022 12:00:00 AM
7/27/2022	07/26/2022	7/26/2022 12:00:00 AM
8/2/2022	08/02/2022	8/2/2022 12:00:00 AM
8/8/2022	08/02/2022	8/2/2022 12:00:00 AM
8/19/2022	08/16/2022	8/16/2022 12:00:00 AM
9/26/2022	09/20/2022	9/20/2022 12:00:00 AM

date	start_of_week	start_of_week_timestamp
10/14/2022	10/11/2022	10/11/2022 12:00:00 AM
10/29/2022	10/25/2022	10/25/2022 12:00:00 AM

Da in diesem Fall das Argument `first_week_date` von 1 in der Funktion `weekstart()` verwendet wird, legt sie Dienstag als ersten Tag der Woche fest.

Diagramm der Funktion `weekstart()`, Beispiel „first_week_day“

Transaktion 8191 fand am 5. Februar statt. Die Funktion `weekstart()` identifiziert, dass der erste Dienstag vor diesem Datum – und somit der Start der Woche und der zurückgegebene Wert – der 1. Februar um 12:00:00 AM war.

Beispiel 4 – Diagrammobjektbeispiel

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript verwendet den gleichen Datensatz und das gleiche Szenario wie das erste Beispiel.

In diesem Beispiel wird jedoch der unveränderte Datensatz in die Anwendung geladen. Die Berechnung, die einen Zeitstempel für den Start der Woche zurückgibt, in dem die Transaktionen stattfanden, wird als Kennzahl in einem Diagrammobjekt der Anwendung erstellt.

Ladeskript

```
Transactions:

Load
*
Inline
[
id,date,amount
8188,1/7/2022,17.17
8189,1/19/2022,37.23
8190,2/28/2022,88.27
8191,2/5/2022,57.42
8192,3/16/2022,53.80
8193,4/1/2022,82.06
8194,5/7/2022,40.39
8195,5/16/2022,87.21
```

```
8196,6/15/2022,95.93
8197,6/26/2022,45.89
8198,7/9/2022,36.23
8199,7/22/2022,25.66
8200,7/23/2022,82.77
8201,7/27/2022,69.98
8202,8/2/2022,76.11
8203,8/8/2022,25.12
8204,8/19/2022,46.23
8205,9/26/2022,84.21
8206,10/14/2022,96.24
8207,10/29/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu: `date`.

Um den Start der Woche zu berechnen, in der eine Transaktion stattfindet, erstellen Sie die folgenden Kennzahlen:

- `=weekstart(date)`
- `=timestamp(weekstart(date))`

Ergebnistabelle

date	start_of_week	start_of_week_timestamp
1/7/2022	01/02/2022	1/2/2022 12:00:00 AM
1/19/2022	01/16/2022	1/16/2022 12:00:00 AM
2/5/2022	01/30/2022	1/30/2022 12:00:00 AM
2/28/2022	02/27/2022	2/27/2022 12:00:00 AM
3/16/2022	03/13/2022	3/13/2022 12:00:00 AM
4/1/2022	03/27/2022	3/27/2022 12:00:00 AM
5/7/2022	05/01/2022	5/1/2022 12:00:00 AM
5/16/2022	05/15/2022	5/15/2022 12:00:00 AM
6/15/2022	06/12/2022	6/12/2022 12:00:00 AM
6/26/2022	06/26/2022	6/26/2022 12:00:00 AM
7/9/2022	07/03/2022	7/3/2022 12:00:00 AM
7/22/2022	07/17/2022	7/17/2022 12:00:00 AM
7/23/2022	07/17/2022	7/17/2022 12:00:00 AM
7/27/2022	07/24/2022	7/24/2022 12:00:00 AM

date	start_of_week	start_of_week_timestamp
8/2/2022	07/31/2022	7/31/2022 12:00:00 AM
8/8/2022	08/07/2022	8/7/2022 12:00:00 AM
8/19/2022	08/14/2022	8/14/2022 12:00:00 AM
9/26/2022	09/25/2022	9/25/2022 12:00:00 AM
10/14/2022	10/09/2022	10/9/2022 12:00:00 AM
10/29/2022	10/23/2022	10/23/2022 12:00:00 AM

Die Kennzahl `start_of_week` wird im Diagrammobjekt erstellt, indem die Funktion `weekstart()` verwendet und das Feld `date` als Argument der Funktion übergeben wird.

Die Funktion `weekstart()` identifiziert zunächst, in welche Woche der Datumswert fällt, und gibt einen Zeitstempel für die erste Millisekunde dieser Woche zurück.

Diagramm der Funktion `weekstart()`, Diagrammobjektbeispiel

Transaktion 8191 fand am 5. Februar statt. Die Systemvariable `FirstWeekDay` legt den ersten Tag der Woche als Sonntag fest. Die Funktion `weekstart()` identifiziert, dass der erste Sonntag vor dem 5. Februar – und damit der Start der Woche – der 30. Januar war. Daher gibt der Wert für `start_of_week` dieser Transaktion die erste Millisekunde dieses Tages zurück, also den 30. Januar um 12:00:00 AM.

Beispiel 5 – Szenario

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Datensatz, der in eine Tabelle namens `Payroll` geladen wird
- Daten bestehend aus Mitarbeiter-IDs, Mitarbeiternamen und dem täglich von jedem Mitarbeiter verdienten Lohn

Mitarbeiter beginnen am Montag mit der Arbeit und arbeiten sechs Tage pro Woche. Die Systemvariable `FirstWeekDay` darf nicht geändert werden.

Der Endbenutzer möchte ein Diagrammobjekt, das nach Mitarbeiter-ID und Mitarbeiternamen den in der Woche bis dato verdienten Lohn anzeigt.

Ladeskript

```
Payroll:
Load
*
Inline
[
employee_id,employee_name,day_rate
182,Mark, $150
183,Deryck, $125
184,Dexter, $125
185,Sydney,$270
186,Agatha,$128
];
```

Ergebnisse

Gehen Sie folgendermaßen vor:

1. Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:
    - `employee_id`
    - `employee_name`
2. Erstellen Sie dann eine Kennzahl, um den in der Woche bis dato verdienten Lohn zu berechnen: `=if(today(1)-weekstart(today(1),0,0)<7,(today(1)-weekstart(today(1),0,0))*day_rate,day_rate*6)`
3. Legen Sie das **Zahlenformat** der Kennzahl auf **Währung** fest.

Ergebnistabelle

employee_id	employee_name	=if(today(1)-weekstart(today(1),0,0)<7,(today(1)-weekstart(today(1),0,0))*day_rate,day_rate*6)
182	Mark	$600.00
183	Deryck	$500.00
184	Dexter	$500.00
185	Sydney	$1080.00
186	Agatha	$512.00

Da die Funktion `weekstart()` das aktuelle Datum als erstes Argument und 0 als drittes Argument verwendet, wird Montag als der erste Tag der Woche festgelegt und das Startdatum der aktuellen Woche zurückgegeben. Die Formel zieht dieses Ergebnis vom aktuellen Datum ab und gibt dann die Anzahl der Tage zurück, die bisher in der Woche verstrichen sind.

Mit der Bedingung wird dann ausgewertet, ob diese Woche mehr als sechs Tage lang war. Wenn dies der Fall ist, wird die `day_rate` des Mitarbeiters mit 6 multipliziert. Andernfalls wird die `day_rate` mit der Anzahl der Tage multipliziert, die bisher in dieser Woche verstrichen sind.

weekyear

Diese Funktion liefert das Jahr, zu dem die Wochennummer gemäß den Umgebungsvariablen zählt. Die Kalenderwochen bewegen sich zwischen 1 und circa 52.

Syntax:

```
weekyear(timestamp [, first_week_day [, broken_weeks [, reference_day]]])
```

Rückgabe Datentyp: ganze Zahl

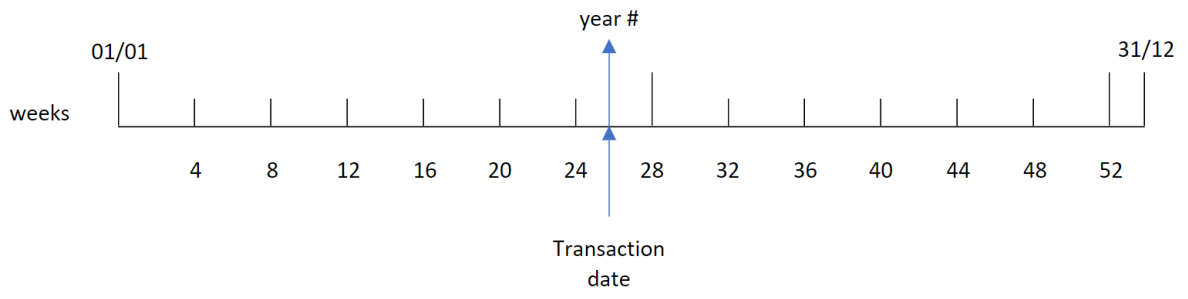
Argumente

Argument	Beschreibung
timestamp	Datum oder Zeitstempel für die Evaluierung.
first_week_day	Legt den Tag fest, an dem die Woche beginnt. Ist nichts definiert, wird der Wert der Variablen **FirstWeekDay** verwendet. Die möglichen Werte für **first_week_day** sind 0 für Montag, 1 für Dienstag, 2 für Mittwoch, 3 für Donnerstag, 4 für Freitag, 5 für Samstag und 6 für Sonntag. Weitere Informationen über die Systemvariable finden Sie unter *FirstWeekDay (page 233)*.
broken_weeks	Wenn Sie **broken_weeks** nicht angeben, wird der Wert der Variablen **BrokenWeeks** verwendet, um festzulegen, ob die Wochen gestückelt sind oder nicht.
reference_day	Wenn Sie **reference_day** nicht angeben, wird der Wert der Variablen **ReferenceDay** verwendet, um festzulegen, welcher Tag im Januar als Referenztag für die Definition von Woche 1 konfiguriert wird. Standardmäßig verwenden Qlik Sense-Funktionen 4 als Referenztag. Das heißt, dass die Woche 1 den 4. Januar enthalten muss, oder anders ausgedrückt: Woche 1 muss immer mindestens 4 Tage im Januar enthalten.

Die Funktion `weekyear()` bestimmt, in welche Woche eines Jahres das Datum fällt. Dann gibt sie das Jahr zurück, das dieser Wochennummer entspricht.

Wenn `BrokenWeeks` auf 0 (false) festgelegt ist, gibt `weekyear()` den gleichen Wert wie `year()` zurück.

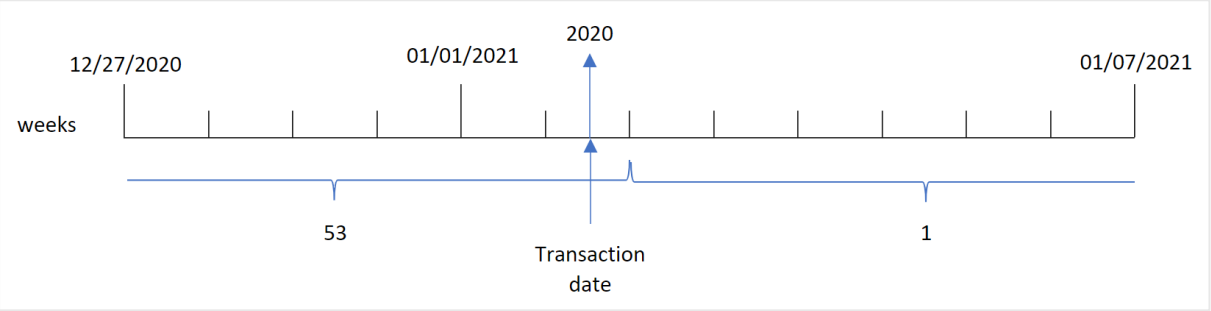
Diagramm des Bereichs der Funktion `weekyear()`



Wenn jedoch die Systemvariable `BrokenWeeks` auf nicht gestückelte Wochen festgelegt ist, darf Woche 1 nur eine bestimmte Anzahl Tage im Januar umfassen, basierend auf dem Wert, der in der Systemvariablen `ReferenceDay` angegeben ist.

Wenn beispielsweise ein Wert für `ReferenceDay` von 4 verwendet wird, muss Woche 1 mindestens vier Tage im Januar enthalten. Woche 1 kann Datumswerte im Dezember des Vorjahres enthalten, oder die letzte Wochennummer eines Jahres kann Datumswerte im Januar des Folgejahres enthalten. In diesen Situationen gibt die Funktion `weekyear()` einen anderen Wert als die Funktion `year()` zurück.

Diagramm des Bereichs der Funktion `weekyear()`, wenn nicht gestückelte Wochen verwendet werden



Verwendung

Die Funktion `weekyear()` ist nützlich, wenn Sie Aggregierungen nach Jahren vergleichen möchten. Das ist beispielsweise der Fall, wenn Sie den Gesamtumsatz von Produkten nach Jahr anzeigen möchten. Die Funktion `weekyear()` wird anstelle von `year()` gewählt, wenn der Benutzer die Einheitlichkeit mit der Systemvariablen `BrokenWeeks` in der App beibehalten möchte.

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Funktionsbeispiele

Beispiel	**Ergebnis**
`weekyear('12/30/1996',0,0,4)`	Gibt 1997 zurück, weil Woche 1 von 1997 am 12/30/1996 beginnt

Beispiel	Ergebnis
`weekyear('01/02/1997',0,0,4)`	Gibt 1997 zurück
`weekyear('12/28/1997',0,0,4)`	Gibt 1997 zurück
`weekyear('12/30/1997',0,0,4)`	Gibt 1998 zurück, weil die Woche 1 von 1998 am 12/29/1997 beginnt
`weekyear('01/02/1999',0,0,4)`	Gibt 1998, weil die Woche 53 von 1998 am 01/03/1999 endet

Verwandte Themen

Thema	Interaktion
week (page 1081)	Gibt eine Ganzzahl für die Wochennummer gemäß ISO 8601 zurück.
year (page 1155)	Gibt das Jahr als Ganzzahl zurück, wenn die Formel entsprechend der Standardzahleninterpretation als Datum interpretiert wird.

Beispiel 1 – Gestückelte Wochen

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Datensatz mit einer Reihe von Transaktionen für die letzte Woche in 2020 und die erste Woche in 2021, der in eine Tabelle namens „`Transactions`“ geladen wird
- Variable `BrokenWeeks`, die auf 1 festgelegt ist
- Ein vorangehender load-Befehl, der Folgendes enthält:
  - Die Funktion `weekyear()`, die als das Feld „`week_year`“ festgelegt ist, das das Jahr zurückgibt, in dem die Transaktion stattfand.
  - Die Funktion `week()`, die als das Feld „`week`“ festgelegt ist, das die Wochennummer für jedes Transaktionsdatum zeigt.

Ladeskript

```
SET BrokenWeeks=1;

Transactions:
       Load
       *,
       week(date) as week,
       weekyear(date) as week_year
       ;
Load
*
Inline
```

```
[
id,date,amount
8176,12/28/2020,19.42
8177,12/29/2020,23.80
8178,12/30/2020,82.06
8179,12/31/2020,40.56
8180,01/01/2021,37.23
8181,01/02/2021,17.17
8182,01/03/2021,88.27
8183,01/04/2021,57.42
8184,01/05/2021,67.42
8185,01/06/2021,23.80
8186,01/07/2021,82.06
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `id`
- `date`
- `week`
- `week_year`

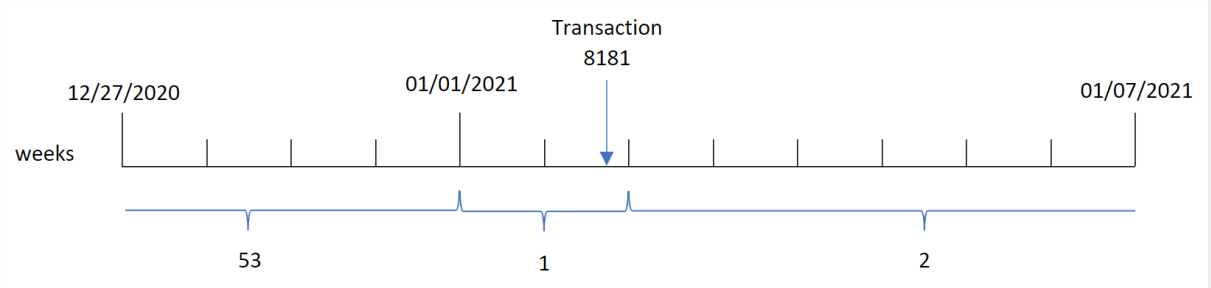
Ergebnistabelle

id	date	week	week_year
8176	12/28/2020	53	2020
8177	12/29/2020	53	2020
8178	12/30/2020	53	2020
8179	12/31/2020	53	2020
8180	01/01/2021	1	2021
8181	01/02/2021	1	2021
8182	01/03/2021	2	2021
8183	01/04/2021	2	2021
8184	01/05/2021	2	2021
8185	01/06/2021	2	2021
8186	01/07/2021	2	2021

Das Feld „week_year“ wird in der vorangehenden load-Anweisung erstellt, indem die Funktion `weekyear()` verwendet und das Datumsfeld als Argument der Funktion übergeben wird.

Die Systemvariable `Brokenweeks` ist auf 1 festgelegt, was bedeutet, dass die App gestückelte Wochen verwendet. Woche 1 beginnt am 1. Januar.

Diagramm des Bereichs der Funktion `weekyear()`, wenn gestückelte Wochen verwendet werden



Transaktion 8181 findet am 2. Januar statt, der in Woche 1 liegt. Daher wird ein Wert von 2021 für das Feld „week_year“ zurückgegeben.

Beispiel 2 – Ungestückelte Wochen

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Datensatz mit einer Reihe von Transaktionen für die letzte Woche in 2020 und die erste Woche in 2021, der in eine Tabelle namens „`Transactions`“ geladen wird
- Variable `Brokenweeks`, die auf 0 festgelegt ist
- Eine vorangehende load-Anweisung, die Folgendes enthält:
  - Die Funktion `weekyear()`, die als das Feld „`week_year`“ festgelegt ist, das das Jahr zurückgibt, in dem die Transaktion stattfand.
  - Die Funktion `week()`, die als das Feld „`week`“ festgelegt ist, das die Wochennummer für jedes Transaktionsdatum zeigt.

In diesem Beispiel legt die Unternehmensrichtlinie aber die Verwendung von nicht gestückelten Wochen fest.

Ladeskript

```
SET BrokenWeeks=0;

Transactions:
       Load
       *,
       week(date) as week,
       weekyear(date) as week_year
       ;
Load
*
Inline
```

```
[
id,date,amount
8176,12/28/2020,19.42
8177,12/29/2020,23.80
8178,12/30/2020,82.06
8179,12/31/2020,40.56
8180,01/01/2021,37.23
8181,01/02/2021,17.17
8182,01/03/2021,88.27
8183,01/04/2021,57.42
8184,01/05/2021,67.42
8185,01/06/2021,23.80
8186,01/07/2021,82.06
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `id`
- `date`
- `week`
- `week_year`

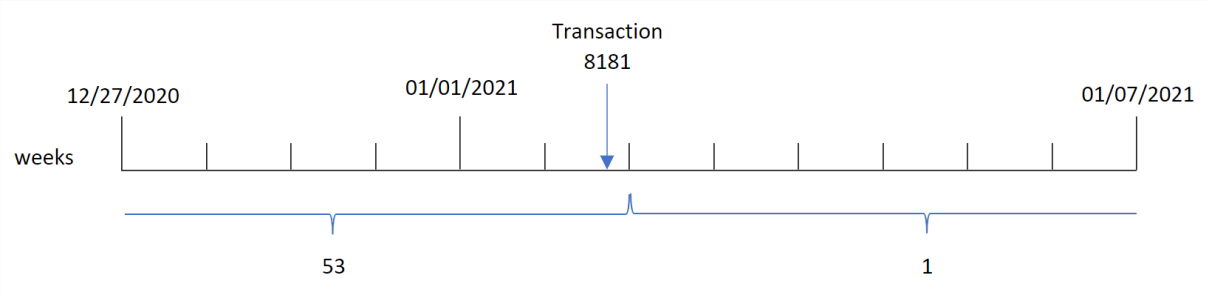
Ergebnistabelle

id	date	week	week_year
8176	12/28/2020	53	2020
8177	12/29/2020	53	2020
8178	12/30/2020	53	2020
8179	12/31/2020	53	2020
8180	01/01/2021	53	2020
8181	01/02/2021	53	2020
8182	01/03/2021	1	2021
8183	01/04/2021	1	2021
8184	01/05/2021	1	2021
8185	01/06/2021	1	2021
8186	01/07/2021	1	2021

Die Systemvariable `Brokenweeks` ist auf `0` festgelegt, was bedeutet, dass die Anwendung nicht gestückelte Wochen verwendet. Daher ist es nicht erforderlich, dass Woche 1 am 1. Januar beginnt.

Woche 53 des Jahres 2020 dauert bis zum Ende des 2. Januars 2021, und Woche 1 von 2021 beginnt am Sonntag, den 3. Januar 2021.

Diagramm des Bereichs der Funktion `weekyear()`, wenn nicht gestückelte Wochen verwendet werden



Transaktion 8181 findet am 2. Januar statt, der in Woche 1 liegt. Daher wird ein Wert von 2021 für das Feld „week_year“ zurückgegeben.

Beispiel 3 – Diagrammobjektbeispiel

Ladeskript und Diagrammformel

Übersicht

Es werden derselbe Datensatz und dasselbe Szenario wie im ersten Beispiel verwendet.

In diesem Beispiel wird jedoch der unveränderte Datensatz in die Anwendung geladen. Die Berechnung, die die Wochennummer des Jahres zurückgibt, in dem die Transaktionen stattfanden, wird als Kennzahl in einem Diagramm der Anwendung erstellt.

Ladeskript

```
SET BrokenWeeks=1;

Transactions:
Load
*
Inline
[
id,date,amount
8176,12/28/2020,19.42
8177,12/29/2020,23.80
8178,12/30/2020,82.06
8179,12/31/2020,40.56
8180,01/01/2021,37.23
8181,01/02/2021,17.17
8182,01/03/2021,88.27
8183,01/04/2021,57.42
8184,01/05/2021,67.42
8185,01/06/2021,23.80
8186,01/07/2021,82.06
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `id`
- `date`

Um die Woche zu berechnen, in der eine Transaktion stattfindet, erstellen Sie die folgende Kennzahl:

- `=week(date)`

Um das Jahr, in dem eine Transaktion stattfindet, basierend auf der Wochennummer zu berechnen, erstellen Sie die folgende Kennzahl:

- `=weekyear(date)`

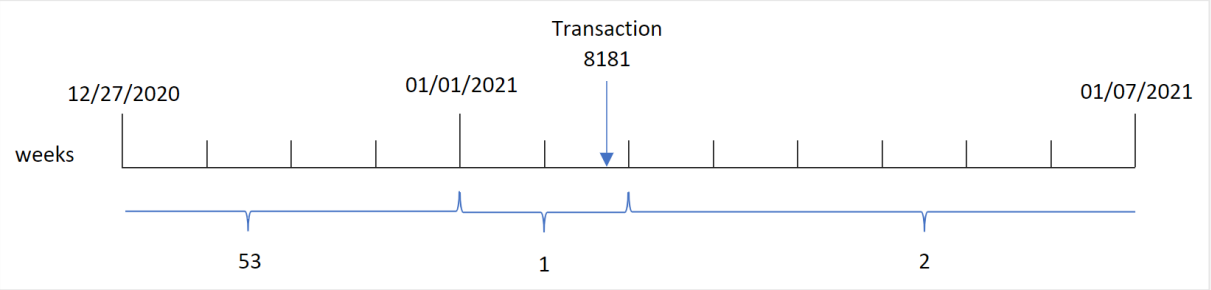
Ergebnistabelle

id	date	week	week_year
8176	12/28/2020	53	2020
8177	12/29/2020	53	2020
8178	12/30/2020	53	2020
8179	12/31/2020	53	2020
8180	01/01/2021	1	2021
8181	01/02/2021	1	2021
8182	01/03/2021	2	2021
8183	01/04/2021	2	2021
8184	01/05/2021	2	2021
8185	01/06/2021	2	2021
8186	01/07/2021	2	2021

Das Feld „`week_year`“ wird in der vorangehenden load-Anweisung erstellt, indem die Funktion `weekyear()` verwendet und das Datumsfeld als Argument der Funktion übergeben wird.

Die Systemvariable `Brokenweeks` ist auf `1` festgelegt, was bedeutet, dass die App gestückelte Wochen verwendet. Woche 1 beginnt am 1. Januar.

Diagramm des Bereichs der Funktion `weekyear()`, wenn gestückelte Wochen verwendet werden



Transaktion 8181 findet am 2. Januar statt, der in Woche 1 liegt. Daher wird ein Wert von 2021 für das Feld „week_year“ zurückgegeben.

Beispiel 4 – Szenario

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Datensatz mit einer Reihe von Transaktionen für die letzte Woche in 2020 und die erste Woche in 2021, der in eine Tabelle namens „`Transactions`“ geladen wird
- Variable `BrokenWeeks`, die auf 0 festgelegt ist Das bedeutet, das die App nicht gestückelte Wochen verwendet.
- Die Variable `ReferenceDay` ist auf 2 festgelegt. Das bedeutet, dass das Jahr am 2. Januar beginnt und mindestens zwei Tage im Januar umfasst.
- Die Variable `FirstWeekDay` ist auf 1 festgelegt. Damit wird der erste Tag der Woche auf einen Dienstag festgelegt.

Gemäß der Unternehmensrichtlinie werden nicht gestückelte Wochen verwendet. Der Endbenutzer möchte ein Diagramm, das den Gesamtumsatz nach Jahr darstellt. Die App verwendet nicht gestückelte Wochen, wobei Woche 1 mindestens zwei Tage im Januar enthält.

Ladeskript

```
SET BrokenWeeks=0;
SET ReferenceDay=2;
SET FirstWeekDay=1;

Transactions:
Load
*
Inline
[
id,date,amount
8176,12/28/2020,19.42
8177,12/29/2020,23.80
```

```
8178,12/30/2020,82.06
8179,12/31/2020,40.56
8180,01/01/2021,37.23
8181,01/02/2021,17.17
8182,01/03/2021,88.27
8183,01/04/2021,57.42
8184,01/05/2021,67.42
8185,01/06/2021,23.80
8186,01/07/2021,82.06
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle.

Um das Jahr, in dem eine Transaktion stattfindet, basierend auf der Wochennummer zu berechnen, erstellen Sie die folgende Kennzahl:

- `=weekyear(date)`

Erstellen Sie die folgende Kennzahl, um den Gesamtumsatz zu berechnen:

- `sum(amount)`

Legen Sie das **Zahlenformat** der Kennzahl auf **Währung** fest.

Ergebnistabelle

weekyear(date)	=sum(amount)
2020	19.42
2021	373.37

year

Diese Funktion liefert das Jahr als ganze Zahl, wenn **expression** entsprechend der Standardinterpretation als Datum interpretiert wird.

Syntax:

```
year(expression)
```

Rückgabe Datentyp: ganze Zahl

Die Funktion `year()` ist sowohl als Skript als auch als Diagrammfunktion verfügbar. Die Funktion gibt das Jahr für eine bestimmtes Datum zurück. Sie wird gewöhnlich verwendet, um ein Jahresfeld als Dimension in einem Master-Kalender zu erstellen.

Verwendung

Die Funktion `year()` ist nützlich, wenn Sie Aggregierungen nach Jahr vergleichen möchten. Sie kann beispielsweise verwendet werden, wenn Sie den Gesamtumsatz von Produkten nach Jahr anzeigen möchten.

Diese Dimensionen können im Ladeskript erstellt werden, indem die Funktion verwendet wird, um ein Feld in einer Master-Kalender-Tabelle zu erstellen. Alternativ kann sie direkt als berechnete Dimension in einem Diagramm verwendet werden.

Funktionsbeispiele

Beispiel	**Ergebnis**
`year( '2012-10-12' )`	liefert 2012
`year( '35648' )`	liefert 1997, da 35648 = 1997-08-06

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Beispiel 1 – DateFormat-Datensatz (Skript)

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Datensatz mit Datumsangaben, der in eine Tabelle mit dem Namen `Master Calendar` geladen wird
- Die `DateFormat`-Standardsystemvariable MM/TT/JJJJ wird verwendet.
- Ein vorangehender load-Befehl, der zum Erstellen eines zusätzlichen Feldes mit dem Namen `year` unter Verwendung der Funktion `year()` verwendet wird.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Master_Calendar:

    Load
        date,
```

```
        year(date) as year
        ;
Load
date
Inline
[
date
12/28/2020
12/29/2020
12/30/2020
12/31/2020
01/01/2021
01/02/2021
01/03/2021
01/04/2021
01/05/2021
01/06/2021
01/07/2021
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `year`

Ergebnistabelle

date	**year**
12/28/2020	2020
12/29/2020	2020
12/30/2020	2020
12/31/2020	2020
01/01/2021	2021
01/02/2021	2021
01/03/2021	2021
01/04/2021	2021
01/05/2021	2021
01/06/2021	2021
01/07/2021	2021

Beispiel 2 – ANSI-Datum

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Datensatz mit Datumsangaben, der in eine Tabelle mit dem Namen `Master Calendar` geladen wird
- Die `DateFormat`-Standardsystemvariable (MM/TT/JJJJ) wird verwendet. Die im Datensatz enthaltenen Datumsangaben weisen aber das ANSI-Standarddatumsformat auf.
- Eine vorangehende load-Anweisung, die zum Erstellen eines zusätzlichen Feldes mit dem Namen `year` unter Verwendung der Funktion `year()` verwendet wird

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Master_Calendar:
    Load
        date,
        year(date) as year
        ;
Load
date
Inline
[
date
2020-12-28
2020-12-29
2020-12-30
2020-12-31
2021-01-01
2021-01-02
2021-01-03
2021-01-04
2021-01-05
2021-01-06
2021-01-07
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `year`

Ergebnistabelle

date	year
2020-12-28	2020
2020-12-29	2020
2020-12-30	2020
2020-12-31	2020
2021-01-01	2021
2021-01-02	2021
2021-01-03	2021
2021-01-04	2021
2021-01-05	2021
2021-01-06	2021
2021-01-07	2021

Beispiel 3 – Unformatiertes Datum

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Datensatz mit Datumsangaben im numerischen Format, der in eine Tabelle mit dem Namen `Master Calendar` geladen wird
- Die `DateFormat`-Standardsystemvariable (MM/TT/JJJJ) wird verwendet.
- Eine vorangehende load-Anweisung, die zum Erstellen eines zusätzlichen Feldes mit dem Namen `year` unter Verwendung der Funktion `year()` verwendet wird

Das ursprüngliche unformatierte Datum wird mit dem Namen `unformatted_date` geladen, und für mehr Klarheit wird ein weiteres Feld mit dem Namen `long_date` verwendet, um das numerische Datum anhand der Funktion `date()` in ein formatiertes Datumsfeld zu konvertieren.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Master_Calendar:
    Load
        unformatted_date,
        date(unformatted_date) as long_date,
        year(unformatted_date) as year
```

```
        ;
Load
unformatted_date
Inline
[
unformatted_date
44868
44898
44928
44958
44988
45018
45048
45078
45008
45038
45068
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `unformatted_date`
- `long_date`
- `year`

Ergebnistabelle

unformatted_date	long_date	year
44868	11/03/2022	2022
44898	12/03/2022	2022
44928	01/02/2023	2023
44958	02/01/2023	2023
44988	03/03/2023	2023
45008	03/23/2023	2023
45018	04/02/2023	2023
45038	04/22/2023	2023
45048	05/02/2023	2023
45068	05/22/2023	2023
45078	06/01/2023	2023

Beispiel 4 – Diagrammobjektbeispiel

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

In diesem Beispiel wird ein Datensatz mit Bestellungen in eine Tabelle namens „Sales“ geladen. Die Tabelle enthält drei Felder:

- `id`
- `sales_date`
- `amount`

Garantien für Produktverkäufe sind für zwei Jahre ab dem Verkaufsdatum gültig. Die Aufgabe besteht darin, eine Kennzahl in einem Diagramm zu erstellen, um das Jahr zu bestimmen, in dem jede Garantie abläuft.

Ladeskript

```
Sales:
Load
id,
sales_date,
amount
Inline
[
id,sales_date,amount
1,12/28/2020,231.24,
2,12/29/2020,567.28,
3,12/30/2020,364.28,
4,12/31/2020,575.76,
5,01/01/2021,638.68,
6,01/02/2021,785.38,
7,01/03/2021,967.46,
8,01/04/2021,287.67
9,01/05/2021,764.45,
10,01/06/2021,875.43,
11,01/07/2021,957.35
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu: `sales_date`.

Erstellen Sie die folgende Kennzahl:

```
=year(sales_date+365*2)
```

Ergebnistabelle

sales_date	=year(sales_date+365*2)
12/28/2020	2022
12/29/2020	2022
12/30/2020	2022
12/31/2020	2022
01/01/2021	2023
01/02/2021	2023
01/03/2021	2023
01/04/2021	2023
01/05/2021	2023
01/06/2021	2023
01/07/2021	2023

Die Ergebnisse dieser Kennzahl sind in der obigen Tabelle dargestellt. Um zwei Jahre zu einem Datum hinzuzufügen, multiplizieren Sie 365 mit 2 und addieren Sie das Ergebnis zum Verkaufsdatum. Daher ist das Ablaufjahr für Verkäufe, die im Jahr 2020 stattfanden, das Jahr 2022.

yearend

Diese Funktion liefert den Zeitstempel der letzten Millisekunde des letzten Tages des Jahres, in dem **date** liegt. Das Ergebnis wird entsprechend dem im Skript definierten **DateFormat** formatiert.

Syntax:

```
YearEnd( date[, period_no[, first_month_of_year = 1]])
```

Die Funktion `yearend()` legt also fest, in welches Jahr das Datum fällt. Sie gibt dann einen Zeitstempel im Datumsformat für die letzte Millisekunde dieses Jahres zurück. Der erste Monat des Jahres ist standardmäßig der Januar. Sie können aber ändern, welcher Monat als erster festgelegt wird, indem Sie das Argument `first_month_of_year` in der Funktion `yearend()` verwenden.

Die Funktion `yearend()` berücksichtigt die Systemvariable `FirstMonthOfYear` nicht. Das Jahr beginnt am 1. Januar, es sei denn, das Argument `first_month_of_year` wird verwendet, um dies zu ändern.

Diagramm der Funktion `yearend()`.

Verwendung

Die Funktion `yearend()` wird als Teil einer Formel verwendet, wenn in der Berechnung der Teil des Jahres verwendet werden soll, der noch nicht eingetreten ist. Beispiel: Sie möchten die gesamten, während des Jahres noch nicht fällig gewordenen Zinsen berechnen.

Rückgabe Datentyp: dual

Argumente

Argument	Beschreibung
date	Datum oder Zeitstempel für die Evaluierung.
period_no	**period_no** ist eine ganze Zahl, wobei 0 für das Jahr steht, das **date** enthält. Negative Werte von **period_no** stehen für vorangehende, positive Werte für nachfolgende Jahre.
first_month_of_year	Wenn Sie mit abweichenden Geschäftsjahren arbeiten möchten, definieren Sie mit einer Zahl zwischen 2 und 12 für **first_month_of_year** einen anderen Startmonat.

Sie können die folgenden Werte verwenden, um den ersten Monat des Jahres im Argument `first_month_of_year` festzulegen.

Werte für first_month_of_year

Monat	Wert
Februar	2
März	3
April	4
Mai	5
Juni	6
Juli	7
August	8
September	9

Monat	Wert
Oktober	10
November	11
Dezember	12

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Funktionsbeispiele

Beispiel	Ergebnis
`yearend('10/19/2001')`	Liefert 12/31/2001 23:59:59.
`yearend('10/19/2001', -1)`	Liefert 12/31/2000 23:59:59.
`yearend('10/19/2001', 0, 4)`	Liefert 03/31/2002 23:59:59.

Beispiel 1 – keine zusätzlichen Argumente

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz, der eine Reihe von Transaktionen zwischen 2020 und 2022 enthält, wird in eine Tabelle namens „`Transactions`“ geladen.
- Das Datumsfeld wurde im Format `DateFormat` der Systemvariablen `(MM/DD/YYYY)` bereitgestellt.
- Ein vorangehender load-Befehl, der Folgendes enthält:

- Funktion `yearend()`, die als Feld `year_end` festgelegt ist.
- Funktion `Timestamp()`, die als Feld `year_end_timestamp` festgelegt ist.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
       *,
       yearend(date) as year_end,
       timestamp(yearend(date)) as year_end_timestamp
       ;
Load
*
Inline
[
id,date,amount
8188,01/13/2020,37.23
8189,02/26/2020,17.17
8190,03/27/2020,88.27
8191,04/16/2020,57.42
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,06/06/2022,46.23
8205,07/18/2022,84.21
8206,11/14/2022,96.24
8207,12/12/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `id`
- `date`
- `year_end`
- `year_end_timestamp`

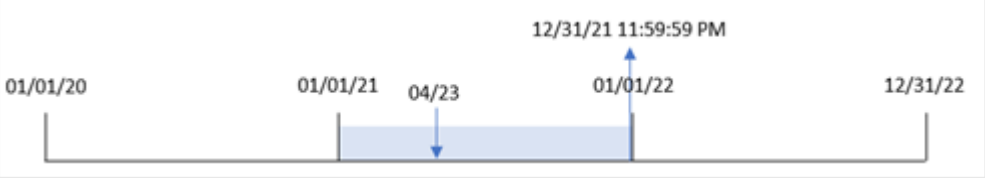
Ergebnistabelle

id	date	year_end	year_end_timestamp
8188	01/13/2020	12/31/2020	12/31/2020 11:59:59 PM
8189	02/26/2020	12/31/2020	12/31/2020 11:59:59 PM
8190	03/27/2020	12/31/2020	12/31/2020 11:59:59 PM
8191	04/16/2020	12/31/2020	12/31/2020 11:59:59 PM
8192	05/21/2020	12/31/2020	12/31/2020 11:59:59 PM
8193	08/14/2020	12/31/2020	12/31/2020 11:59:59 PM
8194	10/07/2020	12/31/2020	12/31/2020 11:59:59 PM
8195	12/05/2020	12/31/2020	12/31/2020 11:59:59 PM
8196	01/22/2021	12/31/2021	12/31/2021 11:59:59 PM
8197	02/03/2021	12/31/2021	12/31/2021 11:59:59 PM
8198	03/17/2021	12/31/2021	12/31/2021 11:59:59 PM
8199	04/23/2021	12/31/2021	12/31/2021 11:59:59 PM
8200	05/04/2021	12/31/2021	12/31/2021 11:59:59 PM
8201	06/30/2021	12/31/2021	12/31/2021 11:59:59 PM
8202	07/26/2021	12/31/2021	12/31/2021 11:59:59 PM
8203	12/27/2021	12/31/2021	12/31/2021 11:59:59 PM
8204	06/06/2022	12/31/2022	12/31/2022 11:59:59 PM
8205	07/18/2022	12/31/2022	12/31/2022 11:59:59 PM
8206	11/14/2022	12/31/2022	12/31/2022 11:59:59 PM
8207	12/12/2022	12/31/2022	12/31/2022 11:59:59 PM

Das Feld „year_end“ wird in der vorangehenden load-Anweisung erstellt, indem die Funktion `yearend()` verwendet und das Datumsfeld als Argument der Funktion übergeben wird.

Die Funktion `yearend()` identifiziert zuerst, in welches Jahr der Datumswert fällt, und gibt einen Zeitstempel für die letzte Millisekunde dieses Jahres zurück.

Diagramm der Funktion `yearend()` mit ausgewählter Transaktion 8199.



Transaktion 8199 fand am 23. April 2021 statt. Die Funktion `yearend()` gibt die letzte Millisekunde dieses Jahres zurück, also den 31. Dezember um 11:59:59 PM.

Beispiel 2 – period_no

Ladeskript und Ergebnisse

Übersicht

Es werden derselbe Datenatz und dasselbe Szenario wie im ersten Beispiel verwendet.

In diesem Beispiel besteht aber die Aufgabe darin, ein Feld „`previous_year_end`“ zu erstellen, das den Enddatumsstempel des Jahres vor dem Jahr zurückgibt, in dem die Transaktion stattfand.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
       *,
       yearend(date,-1) as previous_year_end,
       timestamp(yearend(date,-1)) as previous_year_end_timestamp
       ;
Load
*
Inline
[
id,date,amount
8188,01/13/2020,37.23
8189,02/26/2020,17.17
8190,03/27/2020,88.27
8191,04/16/2020,57.42
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,06/06/2022,46.23
8205,07/18/2022,84.21
8206,11/14/2022,96.24
8207,12/12/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

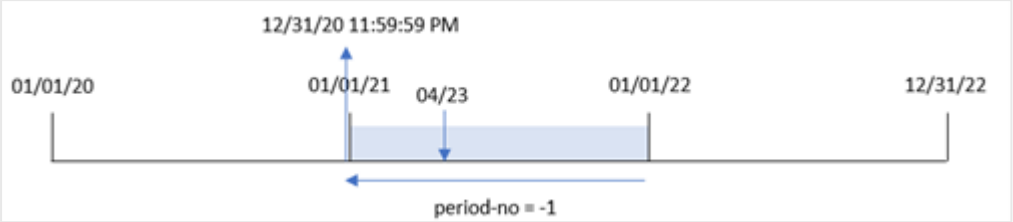
- `id`
- `date`
- `previous_year_end`
- `previous_ year_end_timestamp`

Ergebnistabelle

id	date	previous_year_end	previous_year_end_timestamp
8188	01/13/2020	12/31/2019	12/31/2019 11:59:59 PM
8189	02/26/2020	12/31/2019	12/31/2019 11:59:59 PM
8190	03/27/2020	12/31/2019	12/31/2019 11:59:59 PM
8191	04/16/2020	12/31/2019	12/31/2019 11:59:59 PM
8192	05/21/2020	12/31/2019	12/31/2019 11:59:59 PM
8193	08/14/2020	12/31/2019	12/31/2019 11:59:59 PM
8194	10/07/2020	12/31/2019	12/31/2019 11:59:59 PM
8195	12/05/2020	12/31/2019	12/31/2019 11:59:59 PM
8196	01/22/2021	12/31/2020	12/31/2020 11:59:59 PM
8197	02/03/2021	12/31/2020	12/31/2020 11:59:59 PM
8198	03/17/2021	12/31/2020	12/31/2020 11:59:59 PM
8199	04/23/2021	12/31/2020	12/31/2020 11:59:59 PM
8200	05/04/2021	12/31/2020	12/31/2020 11:59:59 PM
8201	06/30/2021	12/31/2020	12/31/2020 11:59:59 PM
8202	07/26/2021	12/31/2020	12/31/2020 11:59:59 PM
8203	12/27/2021	12/31/2020	12/31/2020 11:59:59 PM
8204	06/06/2022	12/31/2021	12/31/2021 11:59:59 PM
8205	07/18/2022	12/31/2021	12/31/2021 11:59:59 PM
8206	11/14/2022	12/31/2021	12/31/2021 11:59:59 PM
8207	12/12/2022	12/31/2021	12/31/2021 11:59:59 PM

Da eine `period_no` von `-1` als Versatzargument in der Funktion `yearend()` verwendet wurde, identifiziert die Funktion zuerst das Jahr, in dem die Transaktionen stattfinden. Dann geht sie ein Jahr zurück und identifiziert die letzte Millisekunde dieses Jahres.

Diagramm der Funktion `yearend()` mit einem Versatz für „period_no“ von -1.



Transaktion 8199 findet am 23. April 2021 statt. Die Funktion `yearend()` gibt die letzte Millisekunde des vorherigen Jahres, also den 31. Dezember 2020 um 11:59:59 PM. für das Feld „`previous_year_end`“ zurück.

Beispiel 3 – first_month_of_year

Ladeskript und Ergebnisse

Übersicht

Es werden derselbe Datenatz und dasselbe Szenario wie im ersten Beispiel verwendet.

In diesem Beispiel soll aber laut der Unternehmensrichtlinie das Jahr am 1. April beginnen.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
       *,
       yearend(date,0,4) as year_end,
       timestamp(yearend(date,0,4)) as year_end_timestamp
       ;
Load
*
Inline
[
id,date,amount
8188,01/13/2020,37.23
8189,02/26/2020,17.17
8190,03/27/2020,88.27
8191,04/16/2020,57.42
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
```

```
8204,06/06/2022,46.23
8205,07/18/2022,84.21
8206,11/14/2022,96.24
8207,12/12/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

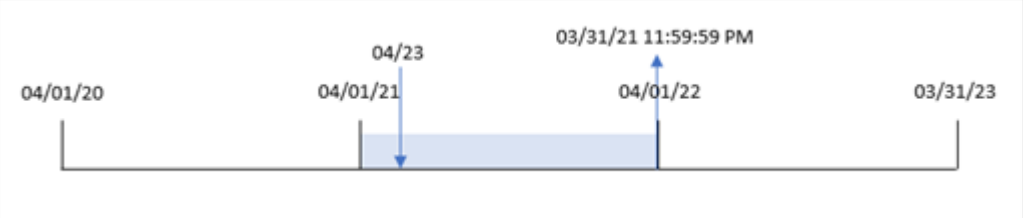
- `id`
- `date`
- `year_end`
- `year_end_timestamp`

Ergebnistabelle

id	date	year_end	year_end_timestamp
8188	01/13/2020	03/31/2020	3/31/2020 11:59:59 PM
8189	02/26/2020	03/31/2020	3/31/2020 11:59:59 PM
8190	03/27/2020	03/31/2020	3/31/2020 11:59:59 PM
8191	04/16/2020	03/31/2021	3/31/2021 11:59:59 PM
8192	05/21/2020	03/31/2021	3/31/2021 11:59:59 PM
8193	08/14/2020	03/31/2021	3/31/2021 11:59:59 PM
8194	10/07/2020	03/31/2021	3/31/2021 11:59:59 PM
8195	12/05/2020	03/31/2021	3/31/2021 11:59:59 PM
8196	01/22/2021	03/31/2021	3/31/2021 11:59:59 PM
8197	02/03/2021	03/31/2021	3/31/2021 11:59:59 PM
8198	03/17/2021	03/31/2021	3/31/2021 11:59:59 PM
8199	04/23/2021	03/31/2022	3/31/2022 11:59:59 PM
8200	05/04/2021	03/31/2022	3/31/2022 11:59:59 PM
8201	06/30/2021	03/31/2022	3/31/2022 11:59:59 PM
8202	07/26/2021	03/31/2022	3/31/2022 11:59:59 PM
8203	12/27/2021	03/31/2022	3/31/2022 11:59:59 PM
8204	06/06/2022	03/31/2023	3/31/2023 11:59:59 PM
8205	07/18/2022	03/31/2023	3/31/2023 11:59:59 PM
8206	11/14/2022	03/31/2023	3/31/2023 11:59:59 PM
8207	12/12/2022	03/31/2023	3/31/2023 11:59:59 PM

Da das Argument `first_month_of_year` von `4` in der Funktion `yearend()` verwendet wird, legt sie den ersten Tag des Jahres auf den 1. April und den letzten Tag des Jahres auf den 31. März fest.

Diagramm der Funktion `yearend()` mit April als erstem Monat des Jahres.



Transaktion 8199 findet am 23. April 2021 statt. Da die Funktion `yearend()` den Beginn des Jahres auf den 1. April festlegt, gibt sie den 31. März 2022 als den Wert „`year_end`“ für die Transaktion zurück.

Beispiel 4 – Diagrammobjektbeispiel

Ladeskript und Diagrammformel

Übersicht

Es werden derselbe Datensatz und dasselbe Szenario wie im ersten Beispiel verwendet.

In diesem Beispiel wird jedoch der unveränderte Datensatz in die Anwendung geladen. Die Berechnung, die den Enddatumzeitstempel für das Jahr zurückgibt, in dem eine Transaktion stattfand, wird als Kennzahl in einem Diagrammobjekt der Anwendung erstellt.

Ladeskript

```
Transactions:
Load
*
Inline
[
id,date,amount
8188,01/13/2020,37.23
8189,02/26/2020,17.17
8190,03/27/2020,88.27
8191,04/16/2020,57.42
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,06/06/2022,46.23
```

```
8205,07/18/2022,84.21
8206,11/14/2022,96.24
8207,12/12/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `id`
- `date`

Um zu berechnen, in welchem Jahr eine Transaktion stattfand, erstellen Sie die folgenden Kennzahlen:

- `=yearend(date)`
- `=timestamp(yearend(date))`

Ergebnistabelle

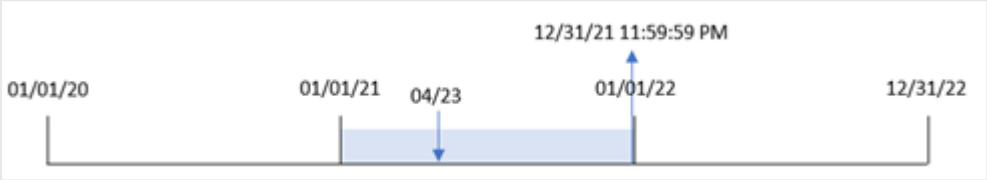
id	date	=yearend(date)	=timestamp(yearend(date))
8188	01/13/2020	12/31/2020	12/31/2020 11:59:59 PM
8189	02/26/2020	12/31/2020	12/31/2020 11:59:59 PM
8190	03/27/2020	12/31/2020	12/31/2020 11:59:59 PM
8191	04/16/2020	12/31/2020	12/31/2020 11:59:59 PM
8192	05/21/2020	12/31/2020	12/31/2020 11:59:59 PM
8193	08/14/2020	12/31/2020	12/31/2020 11:59:59 PM
8194	10/07/2020	12/31/2020	12/31/2020 11:59:59 PM
8195	12/05/2020	12/31/2020	12/31/2020 11:59:59 PM
8196	01/22/2021	12/31/2021	12/31/2021 11:59:59 PM
8197	02/03/2021	12/31/2021	12/31/2021 11:59:59 PM
8198	03/17/2021	12/31/2021	12/31/2021 11:59:59 PM
8199	04/23/2021	12/31/2021	12/31/2021 11:59:59 PM
8200	05/04/2021	12/31/2021	12/31/2021 11:59:59 PM
8201	06/30/2021	12/31/2021	12/31/2021 11:59:59 PM
8202	07/26/2021	12/31/2021	12/31/2021 11:59:59 PM
8203	12/27/2021	12/31/2021	12/31/2021 11:59:59 PM
8204	06/06/2022	12/31/2022	12/31/2022 11:59:59 PM
8205	07/18/2022	12/31/2022	12/31/2022 11:59:59 PM

id	date	=yearend(date)	=timestamp(yearend(date))
8206	11/14/2022	12/31/2022	12/31/2022 11:59:59 PM
8207	12/12/2022	12/31/2022	12/31/2022 11:59:59 PM

Die Kennzahl „end_of_year“ wird im Diagrammobjekt erstellt, indem die Funktion `yearend()` verwendet und das Feld als Argument der Funktion übergeben wird.

Die Funktion `yearend()` identifiziert zuerst, in welches Jahr der Datumswert fällt, und gibt einen Zeitstempel für die letzte Millisekunde dieses Jahres zurück.

Diagramm der Funktion `yearend()`, die zeigt, dass die Transaktion 8199 im April stattfand.



Transaktion 8199 findet am 23. April 2021 statt. Die Funktion `yearend()` gibt die letzte Millisekunde dieses Jahres zurück, also den 31. Dezember um 11:59:59 PM.

Beispiel 5 – Szenario

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz wird in eine Tabelle namens „`Employee_Expenses`“ geladen. Die Tabelle enthält die folgenden Felder:
    - Mitarbeiter-IDs
    - Mitarbeitername
    - Durchschnittliche tägliche Spesenanträge pro Mitarbeiter

Der Endbenutzer möchte ein Diagrammobjekt, das nach Mitarbeiter-ID und Mitarbeitername die geschätzten Spesenanträge anzeigt, die für das restliche Jahr noch anfallen. Das Geschäftsjahr beginnt im Januar.

Ladeskript

```
Employee_Expenses:
Load
*
Inline
[
employee_id,employee_name,avg_daily_claim
182,Mark, $15
```

```
183,Deryck, $12.5
184,Dexter, $12.5
185,Sydney,$27
186,Agatha,$18
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `employee_id`
- `employee_name`

Erstellen Sie die folgende Kennzahl, um die voraussichtlichen Spesenanträge zu berechnen:

`=(yearend(today(1))-today(1))*avg_daily_claim`
Legen Sie die **Zahlenformatierung** der Kennzahl auf **Währung** fest.

Ergebnistabelle

employee_id	employee_name	=(yearend(today(1))-today(1))*avg_daily_claim
182	Mark	$3240.00
183	Deryck	$2700.00
184	Dexter	$2700.00
185	Sydney	$5832.00
186	Agatha	$3888.00

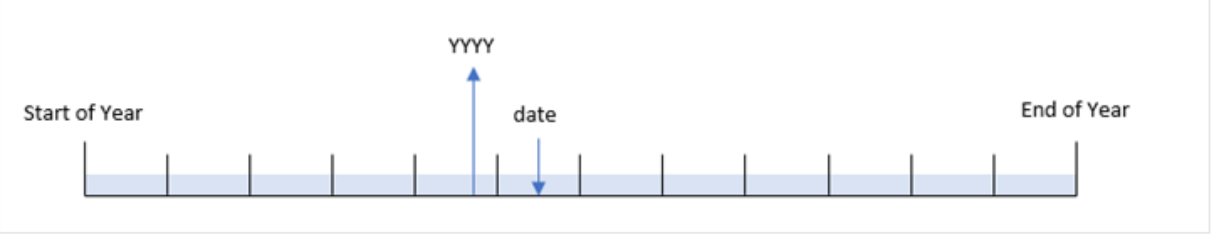
Indem das aktuelle Datum als einziges Argument verwendet wird, gibt die Funktion `yearend()` das Enddatum des aktuellen Jahres zurück. Dann zieht die Formel das aktuelle Datum vom Jahresenddatum ab und gibt die Anzahl der in diesem Jahr verbleibenden Tage zurück.

Dieser Wert wird dann mit den durchschnittlichen täglichen Spesenanträgen der einzelnen Mitarbeitern multipliziert, um den geschätzten Spesenbetrag pro Mitarbeiter für das verbleibende Jahr zu berechnen.

yearname

Diese Funktion liefert den Zeitstempel der ersten Millisekunde des ersten Tages des Jahres, in dem **date** liegt. Das Ergebnis wird als vierstellige Jahreszahl formatiert.

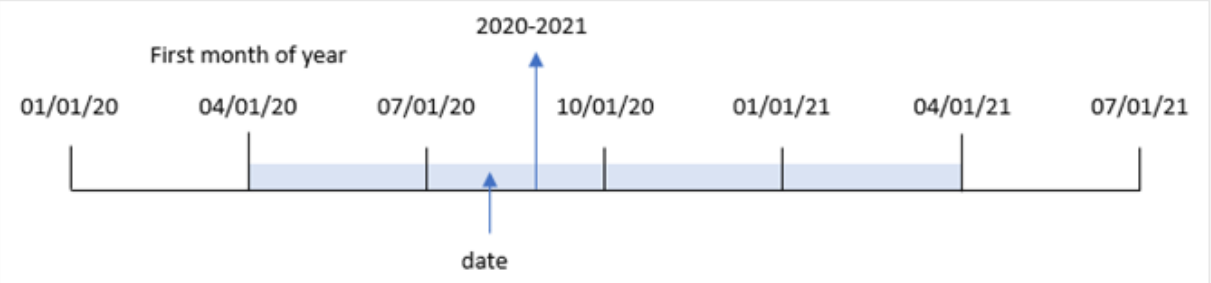
Diagramm des Zeitraums der Funktion `yearname()`.



Die Funktion `yearname()` unterscheidet sich von der Funktion `year()` dahingehend, dass sie einen Versatz des Datums ermöglicht, das ausgewertet werden soll. Außerdem kann der erste Monat des Jahres festgelegt werden.

Wenn der erste Monat des Jahres nicht Januar ist, gibt die Funktion die beiden vierstelligen Jahreszahlen über den Zwölfmonatszeitraum zurück, die das Datum enthalten. Wenn beispielsweise das Jahr im April beginnt und das ausgewertete Datum 06/30/2020 ist, wird als Ergebnis `2020-2021` zurückgegeben.

Diagramm der Funktion `yearname()` mit April als erstem Monat des Jahres.



Syntax:

```
YearName(date[, period_no[, first_month_of_year]] )
```

Rückgabe Datentyp: dual

Argument	Beschreibung
date	Datum oder Zeitstempel für die Evaluierung.
period_no	**period_no** ist eine ganze Zahl, wobei 0 für das Jahr steht, das **date** enthält. Negative Werte von **period_no** stehen für vorangehende, positive Werte für nachfolgende Jahre.
first_month_of_year	Wenn Sie mit abweichenden Geschäftsjahren arbeiten möchten, definieren Sie mit einer Zahl zwischen 2 und 12 für **first_month_of_year** einen anderen Startmonat. In diesem Fall besteht das Ergebnis aus zwei Jahreszahlen.

Sie können die folgenden Werte verwenden, um den ersten Monat des Jahres im Argument `first_month_of_year` festzulegen.

Werte für first_month_of_year

Monat	Wert
Februar	2
März	3
April	4
Mai	5
Juni	6

Monat	Wert
Juli	7
August	8
September	9
Oktober	10
November	11
Dezember	12

Verwendung

Die Funktion `yearname()` eignet sich für den Vergleich von Aggregierungen nach Jahr. Das ist beispielsweise der Fall, wenn Sie den Gesamtumsatz von Produkten nach Jahr anzeigen möchten.

Diese Dimensionen können im Ladeskript erstellt werden, indem die Funktion verwendet wird, um ein Feld in einer Master-Kalender-Tabelle zu erstellen. Sie können auch als berechnete Dimensionen in einem Diagramm erstellt werden.

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Funktionsbeispiele

Beispiel	Ergebnis
`yearname('10/19/2001')`	Gibt „2001“ zurück.
`yearname('10/19/2001',-1)`	Gibt „2000“ zurück.
`yearname('10/19/2001',0,4)`	Gibt „2001-2002“ zurück.

Verwandte Themen

Thema	Beschreibung
year (page 1155)	Diese Funktion liefert das Jahr als ganze Zahl, wenn die Formel entsprechend der Standardzahleninterpretation als Datum interpretiert wird.

Beispiel 1 – keine zusätzlichen Argumente

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz, der eine Reihe von Transaktionen zwischen 2020 und 2022 enthält, wird in eine Tabelle namens „`Transactions`“ geladen.
- Die Systemvariable `DateFormat`, die auf „`MM/DD/YYYY`“ festgelegt ist.
- Ein vorangehender load-Befehl, der die Funktion `yearname()` verwendet und als das Feld `year_name` festgelegt ist.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
      *,
      yearname(date) as year_name
      ;
Load
*
Inline
[
id,date,amount
8188,'01/13/2020',37.23
8189,'02/26/2020',17.17
8190,'03/27/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'08/14/2020',82.06
8194,'10/07/2020',40.39
8195,'12/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
```

```
8202,'07/26/2021',76.11
8203,'12/27/2021',25.12
8204,'06/06/2022',46.23
8205,'07/18/2022',84.21
8206,'11/14/2022',96.24
8207,'12/12/2022',67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `year_name`

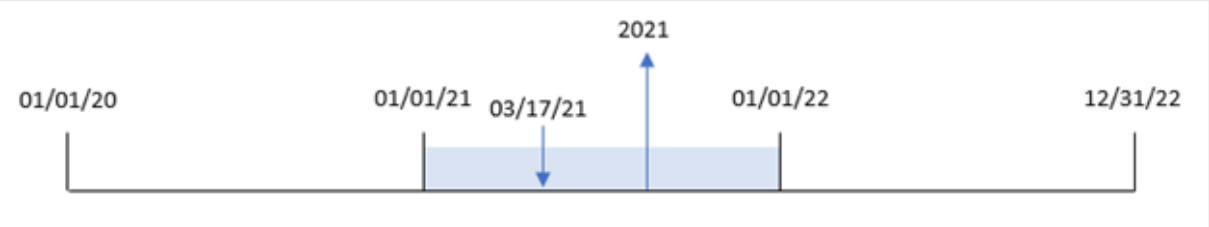
Ergebnistabelle

date	year_name
01/13/2020	2020
02/26/2020	2020
03/27/2020	2020
04/16/2020	2020
05/21/2020	2020
08/14/2020	2020
10/07/2020	2020
12/05/2020	2020
01/22/2021	2021
02/03/2021	2021
03/17/2021	2021
04/23/2021	2021
05/04/2021	2021
06/30/2021	2021
07/26/2021	2021
12/27/2021	2021
06/06/2022	2022
07/18/2022	2022
11/14/2022	2022
12/12/2022	2022

Das Feld „year_name“ wird in der vorangehenden load-Anweisung erstellt, indem die Funktion yearname() verwendet und das Datumsfeld als Argument der Funktion übergeben wird.

Die Funktion yearname() identifiziert, in welches Jahr der Datumswert fällt, und gibt dies als vierstelligen Jahreswert zurück.

Diagramm der Funktion yearname(), die 2021 als den Jahreswert anzeigt



Beispiel 2 – period_no

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz, der eine Reihe von Transaktionen zwischen 2020 und 2022 enthält, wird in eine Tabelle namens „Transactions“ geladen.
- Die Systemvariable „DateFormat“, die auf „MM/DD/YYYY“ festgelegt ist.
- Eine vorangehende load-Anweisung, die die Funktion yearname() verwendet und als das Feld year_name festgelegt ist.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
      *,
      yearname(date,-1) as prior_year_name
      ;
Load
*
Inline
[
id,date,amount
8188,'01/13/2020',37.23
8189,'02/26/2020',17.17
8190,'03/27/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'08/14/2020',82.06
```

```
8194,'10/07/2020',40.39
8195,'12/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'12/27/2021',25.12
8204,'06/06/2022',46.23
8205,'07/18/2022',84.21
8206,'11/14/2022',96.24
8207,'12/12/2022',67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `prior_year_name`

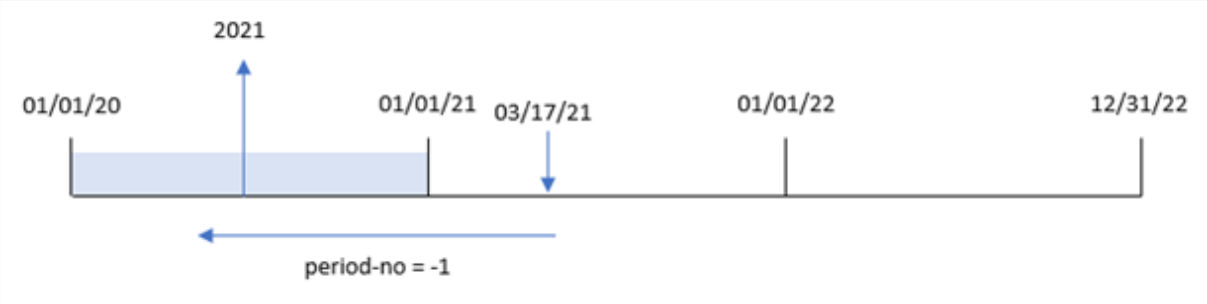
Ergebnistabelle

date	prior_year_name
01/13/2020	2019
02/26/2020	2019
03/27/2020	2019
04/16/2020	2019
05/21/2020	2019
08/14/2020	2019
10/07/2020	2019
12/05/2020	2019
01/22/2021	2020
02/03/2021	2020
03/17/2021	2020
04/23/2021	2020
05/04/2021	2020
06/30/2021	2020
07/26/2021	2020

date	prior_year_name
12/27/2021	2020
06/06/2022	2021
07/18/2022	2021
11/14/2022	2021
12/12/2022	2021

Da eine `period_no` von `-1` als Versatzargument in der Funktion `yearname()` verwendet wird, identifiziert die Funktion zuerst das Jahr, in dem die Transaktionen stattfinden. Die Funktion wechselt dann zum vorherigen Jahr und gibt dieses Jahr als Ergebnis zurück.

Diagramm der Funktion `yearname()`, bei der `period_no` als `-1` festgelegt ist



Beispiel 3 – first_month_of_year

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Den gleichen Datensatz wie im ersten Beispiel.
- Die Systemvariable `DateFormat`, die auf „`MM/DD/YYYY`“ festgelegt ist.
- Eine vorangehende load-Anweisung, die die Funktion `yearname()` verwendet und als das Feld `year_name` festgelegt ist.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
      *,
      yearname(date,0,4) as year_name
      ;
Load
```

```
*
Inline
[
id,date,amount
8188,'01/13/2020',37.23
8189,'02/26/2020',17.17
8190,'03/27/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'08/14/2020',82.06
8194,'10/07/2020',40.39
8195,'12/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'12/27/2021',25.12
8204,'06/06/2022',46.23
8205,'07/18/2022',84.21
8206,'11/14/2022',96.24
8207,'12/12/2022',67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `id`
- `date`
- `year_name`

Ergebnistabelle

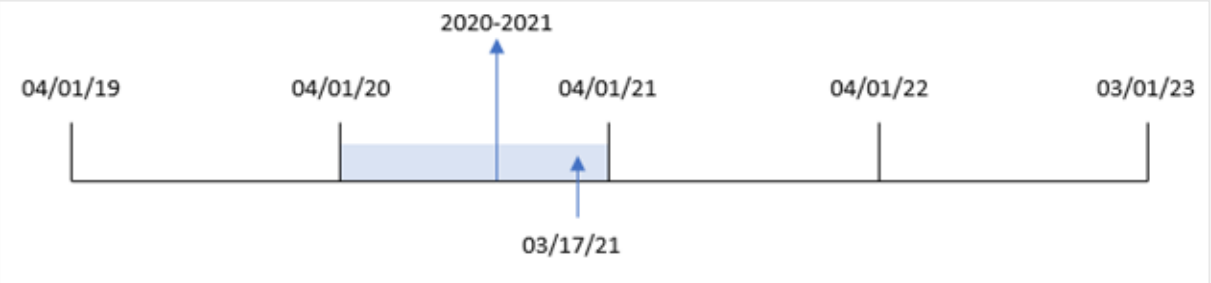
date	year_name
01/13/2020	2019-2020
02/26/2020	2019-2020
03/27/2020	2019-2020
04/16/2020	2020-2021
05/21/2020	2020-2021
08/14/2020	2020-2021
10/07/2020	2020-2021
12/05/2020	2020-2021

date	year_name
01/22/2021	2020-2021
02/03/2021	2020-2021
03/17/2021	2020-2021
04/23/2021	2021-2022
05/04/2021	2021-2022
06/30/2021	2021-2022
07/26/2021	2021-2022
12/27/2021	2021-2022
06/06/2022	2022-2023
07/18/2022	2022-2023
11/14/2022	2022-2023
12/12/2022	2022-2023

Da das Argument `first_month_of_year` von `4` in der Funktion `yearname()` verwendet wird, verschiebt sich der Jahresbeginn vom 1. Januar zum 1. April. Daher liegt jeder Zwölfmonatszeitraum zwischen zwei Kalenderjahren, und die Funktion `yearname()` gibt zwei vierstellige Jahreszahlen für jedes ausgewertete Datum zurück.

Transaktion 8198 findet am 17. März 2021 statt. Die Funktion `yearname()` legt den Jahresbeginn auf den 1. April und das Jahresende auf den 30. März fest. Infolgedessen fand die Transaktion 8198 im Jahreszeitraum vom 1. April 2020 bis zum 30. März 2021 statt. Somit gibt die Funktion `yearname()` den Wert 2020-2021 zurück.

Diagramm der Funktion `yearname()` mit März als erstem Monat des Jahres



Beispiel 4 – Diagrammobjektbeispiel

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Den gleichen Datensatz wie im ersten Beispiel.
- Die Systemvariable `DateFormat`, die auf „`MM/DD/YYYY`“ festgelegt ist.

Das Feld, das das Jahr zurückgibt, in dem die Transaktion stattfand, wird jedoch als Kennzahl in einem Diagrammobjekt erstellt.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8188,'01/13/2020',37.23
8189,'02/26/2020',17.17
8190,'03/27/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'08/14/2020',82.06
8194,'10/07/2020',40.39
8195,'12/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'12/27/2021',25.12
8204,'06/06/2022',46.23
8205,'07/18/2022',84.21
8206,'11/14/2022',96.24
8207,'12/12/2022',67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu:

```
date
```

Um das Feld „`year_name`“ zu berechnen, erstellen Sie die folgende Kennzahl:

```
=yearname(date)
```

Ergebnistabelle

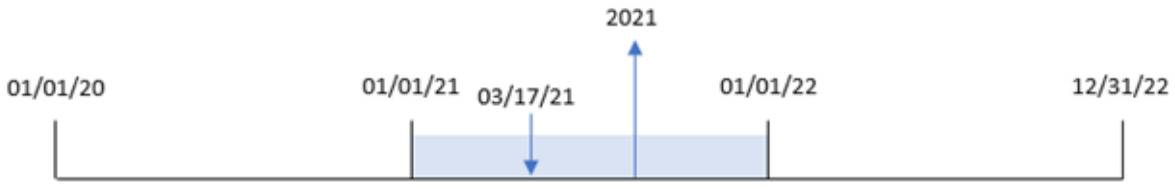
date	=yearname(date)
01/13/2020	2020

date	=yearname(date)
02/26/2020	2020
03/27/2020	2020
04/16/2020	2020
05/21/2020	2020
08/14/2020	2020
10/07/2020	2020
12/05/2020	2020
01/22/2021	2021
02/03/2021	2021
03/17/2021	2021
04/23/2021	2021
05/04/2021	2021
06/30/2021	2021
07/26/2021	2021
12/27/2021	2021
06/06/2022	2022
07/18/2022	2022
11/14/2022	2022
12/12/2022	2022

Die Kennzahl „year_name“ wird im Diagrammobjekt erstellt, indem die Funktion `yearname()` verwendet und das Feld als Argument der Funktion übergeben wird.

Die Funktion `yearname()` identifiziert, in welches Jahr der Datumswert fällt, und gibt dies als vierstelligen Jahreswert zurück.

Diagramm der Funktion `yearname()` mit 2021 als Jahreswert



Beispiel 5 – Szenario

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Den gleichen Datensatz wie im ersten Beispiel.
- Die Systemvariable `DateFormat`, die auf „`MM/DD/YYYY`“ festgelegt ist.

Der Endbenutzer möchte ein Diagramm, das den Gesamtumsatz nach Quartal für die Transaktionen darstellt. Verwenden Sie die Funktion `yearname()` als berechnete Dimension, um dieses Diagramm zu erstellen, wenn die Dimension `yearname()` im Datenmodell nicht verfügbar ist.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8188,'01/13/2020',37.23
8189,'02/26/2020',17.17
8190,'03/27/2020',88.27
8191,'04/16/2020',57.42
8192,'05/21/2020',53.80
8193,'08/14/2020',82.06
8194,'10/07/2020',40.39
8195,'12/05/2020',87.21
8196,'01/22/2021',95.93
8197,'02/03/2021',45.89
8198,'03/17/2021',36.23
8199,'04/23/2021',25.66
8200,'05/04/2021',82.77
8201,'06/30/2021',69.98
8202,'07/26/2021',76.11
8203,'12/27/2021',25.12
8204,'06/06/2022',46.23
8205,'07/18/2022',84.21
8206,'11/14/2022',96.24
8207,'12/12/2022',67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle.

Erstellen Sie die folgende berechnete Dimension, um Aggregierungen nach Jahr zu vergleichen:

```
=yearname(date)
```

Erstellen Sie die folgende Kennzahl:

```
=sum(amount)
```

Legen Sie die **Zahlenformatierung** der Kennzahl auf **Währung** fest.

Ergebnistabelle

yearname(date)	=sum(amount)
2020	$463.55
2021	$457.69
2022	$294.35

yearstart

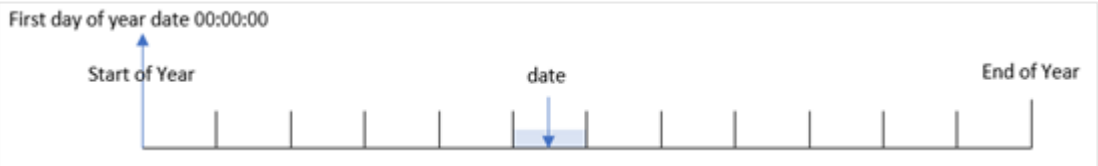
Diese Funktion liefert einen Zeitstempel, der dem Beginn des ersten Jahres mit **date** entspricht. Das Ergebnis wird entsprechend dem im Skript definierten **DateFormat** formatiert.

Syntax:

YearStart(date[, period_no[, first_month_of_year]]**)**

Die Funktion `yearstart()` legt also fest, in welches Jahr das Datum fällt. Sie gibt dann einen Zeitstempel im Datumsformat für die erste Millisekunde dieses Jahres zurück. Der erste Monat des Jahres ist standardmäßig der Januar. Sie können aber ändern, welcher Monat als erster festgelegt wird, indem Sie das Argument `first_month_of_year` in der Funktion `yearstart()` verwenden.

Diagramm der Funktion `yearstart()`, die den Zeitraum zeigt, der von der Funktion abgedeckt werden kann.



Verwendung

Die Funktion `yearstart()` wird als Teil einer Formel verwendet, wenn in der Berechnung der Teil des Jahres verwendet werden soll, der bisher verstrichen ist. Beispiel: Sie möchten die Zinsen berechnen, die in einem Jahr bis dato aufgelaufen sind.

Rückgabe Datentyp: dual

Argumente

Argument	Beschreibung
date	Datum oder Zeitstempel für die Evaluierung.

Argument	Beschreibung
period_no	**period_no** ist eine ganze Zahl, wobei 0 für das Jahr steht, das **date** enthält. Negative Werte von **period_no** stehen für vorangehende, positive Werte für nachfolgende Jahre.
first_month_of_year	Wenn Sie mit abweichenden Geschäftsjahren arbeiten möchten, definieren Sie mit einer Zahl zwischen 2 und 12 für **first_month_of_year** einen anderen Startmonat.

Die folgenden Monate können im `first_month_of_year argument` verwendet werden:

Werte für first_month_of_year

Monat	Wert
Februar	2
März	3
April	4
Mai	5
Juni	6
Juli	7
August	8
September	9
Oktober	10
November	11
Dezember	12

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Funktionsbeispiele

Beispiel	Ergebnis
`yearstart('10/19/2001')`	Gibt 01/01/2001 00:00:00 zurück.
`yearstart('10/19/2001',-1)`	Gibt 01/01/2000 00:00:00 zurück.
`yearstart('10/19/2001',0,4)`	Gibt 04/01/2001 00:00:00 zurück.

Beispiel 1 – einfaches Beispiel

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz, der eine Reihe von Transaktionen zwischen 2020 und 2022 enthält, wird in eine Tabelle namens „`Transactions`“ geladen.
- Das Datumsfeld wurde im Format `DateFormat` der Systemvariablen `MM/DD/YYYY` bereitgestellt.
- Ein vorangehender load-Befehl, der Folgendes enthält:
    - Funktion `yearstart()`, die als Feld `year_start` festgelegt ist.
    - Funktion `Timestamp()`, die als Feld `year_start_timestamp` festgelegt ist.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
       *,
       yearstart(date) as year_start,
       timestamp(yearstart(date)) as year_start_timestamp
       ;
Load
*
Inline
[
id,date,amount
8188,01/13/2020,37.23
8189,02/26/2020,17.17
8190,03/27/2020,88.27
8191,04/16/2020,57.42
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
```

```
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,06/06/2022,46.23
8205,07/18/2022,84.21
8206,11/14/2022,96.24
8207,12/12/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `id`
- `date`
- `year_start`
- `year_start_timestamp`

Ergebnistabelle

id	date	year_start	year_start_timestamp
8188	01/13/2020	01/01/2020	1/1/2020 12:00:00 AM
8189	02/26/2020	01/01/2020	1/1/2020 12:00:00 AM
8190	03/27/2020	01/01/2020	1/1/2020 12:00:00 AM
8191	04/16/2020	01/01/2020	1/1/2020 12:00:00 AM
8192	05/21/2020	01/01/2020	1/1/2020 12:00:00 AM
8193	08/14/2020	01/01/2020	1/1/2020 12:00:00 AM
8194	10/07/2020	01/01/2020	1/1/2020 12:00:00 AM
8195	12/05/2020	01/01/2020	1/1/2020 12:00:00 AM
8196	01/22/2021	01/01/2021	1/1/2021 12:00:00 AM
8197	02/03/2021	01/01/2021	1/1/2021 12:00:00 AM
8198	03/17/2021	01/01/2021	1/1/2021 12:00:00 AM
8199	04/23/2021	01/01/2021	1/1/2021 12:00:00 AM
8200	05/04/2021	01/01/2021	1/1/2021 12:00:00 AM
8201	06/30/2021	01/01/2021	1/1/2021 12:00:00 AM
8202	07/26/2021	01/01/2021	1/1/2021 12:00:00 AM
8203	12/27/2021	01/01/2021	1/1/2021 12:00:00 AM

id	date	year_start	year_start_timestamp
8204	06/06/2022	01/01/2022	1/1/2022 12:00:00 AM
8205	07/18/2022	01/01/2022	1/1/2022 12:00:00 AM
8206	11/14/2022	01/01/2022	1/1/2022 12:00:00 AM
8207	12/12/2022	01/01/2022	1/1/2022 12:00:00 AM

Das Feld „year_start“ wird in der vorangehenden load-Anweisung erstellt, indem die Funktion `yearstart()` verwendet und das Datumsfeld als Argument der Funktion übergeben wird.

Die Funktion `yearstart()` identifiziert zuerst, in welches Jahr der Datumswert fällt, und gibt einen Zeitstempel für die erste Millisekunde dieses Jahres zurück.

Diagramm von Funktion `yearstart()` und Transaktion 8199

Transaktion 8199 fand am 23. April 2021 statt. Die Funktion `yearstart()` gibt die erste Millisekunde dieses Jahres zurück, also den 1. Januar um 12:00:00 AM.

Beispiel 2 – period_no

Ladeskript und Ergebnisse

Übersicht

Es werden derselbe Datenatz und dasselbe Szenario wie im ersten Beispiel verwendet.

In diesem Beispiel besteht aber die Aufgabe darin, ein Feld „previous_year_start“ zu erstellen, das den Startdatumsstempel des Jahres vor dem Jahr zurückgibt, in dem die Transaktion stattfand.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
       *,
       yearstart(date,-1) as previous_year_start,
       timestamp(yearstart(date,-1)) as previous_year_start_timestamp
       ;
Load
*
Inline
[
id,date,amount
```

```
8188,01/13/2020,37.23
8189,02/26/2020,17.17
8190,03/27/2020,88.27
8191,04/16/2020,57.42
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,06/06/2022,46.23
8205,07/18/2022,84.21
8206,11/14/2022,96.24
8207,12/12/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `id`
- `date`
- `previous_year_start`
- `previous_ year_start_timestamp`

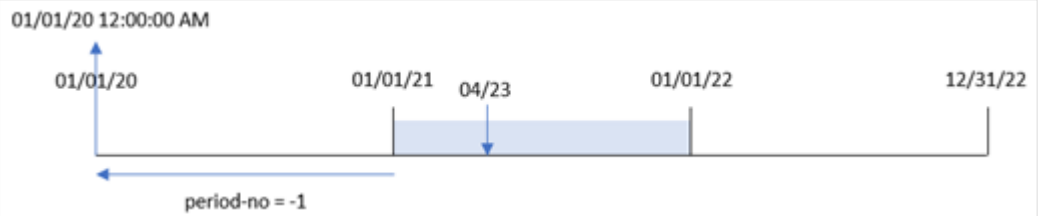
Ergebnistabelle

id	date	previous_year_start	previous_year_start_timestamp
8188	01/13/2020	01/01/2019	1/1/2019 12:00:00 AM
8189	02/26/2020	01/01/2019	1/1/2019 12:00:00 AM
8190	03/27/2020	01/01/2019	1/1/2019 12:00:00 AM
8191	04/16/2020	01/01/2019	1/1/2019 12:00:00 AM
8192	05/21/2020	01/01/2019	1/1/2019 12:00:00 AM
8193	08/14/2020	01/01/2019	1/1/2019 12:00:00 AM
8194	10/07/2020	01/01/2019	1/1/2019 12:00:00 AM
8195	12/05/2020	01/01/2019	1/1/2019 12:00:00 AM
8196	01/22/2021	01/01/2020	1/1/2020 12:00:00 AM
8197	02/03/2021	01/01/2020	1/1/2020 12:00:00 AM

id	date	previous_year_start	previous_year_start_timestamp
8198	03/17/2021	01/01/2020	1/1/2020 12:00:00 AM
8199	04/23/2021	01/01/2020	1/1/2020 12:00:00 AM
8200	05/04/2021	01/01/2020	1/1/2020 12:00:00 AM
8201	06/30/2021	01/01/2020	1/1/2020 12:00:00 AM
8202	07/26/2021	01/01/2020	1/1/2020 12:00:00 AM
8203	12/27/2021	01/01/2020	1/1/2020 12:00:00 AM
8204	06/06/2022	01/01/2021	1/1/2021 12:00:00 AM
8205	07/18/2022	01/01/2021	1/1/2021 12:00:00 AM
8206	11/14/2022	01/01/2021	1/1/2021 12:00:00 AM
8207	12/12/2022	01/01/2021	1/1/2021 12:00:00 AM

Da in dieser Instanz eine `period_no` von `-1` als Versatzargument in der Funktion `yearstart()` verwendet wird, identifiziert die Funktion zuerst das Jahr, in dem die Transaktionen stattfinden. Dann geht sie ein Jahr zurück und identifiziert die erste Millisekunde dieses Jahres.

Diagramm der Funktion `yearstart()` mit einer `period_no` von -1



Transaktion 8199 fand am 23. April 2021 statt. Die Funktion `yearstart()` gibt die erste Millisekunde des vorherigen Jahres, also den 1. Januar 2020 um 12:00:00 AM, für das Feld „`previous_year_start`“ zurück.

Beispiel 3 – first_month_of_year

Ladeskript und Ergebnisse

Übersicht

Es werden derselbe Datenatz und dasselbe Szenario wie im ersten Beispiel verwendet.

In diesem Beispiel soll aber laut der Unternehmensrichtlinie das Jahr am 1. April beginnen.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
       *,
```

```
       yearstart(date,0,4) as year_start,
       timestamp(yearstart(date,0,4)) as year_start_timestamp
       ;
Load
*
Inline
[
id,date,amount
8188,01/13/2020,37.23
8189,02/26/2020,17.17
8190,03/27/2020,88.27
8191,04/16/2020,57.42
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,06/06/2022,46.23
8205,07/18/2022,84.21
8206,11/14/2022,96.24
8207,12/12/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `id`
- `date`
- `year_start`
- `year_start_timestamp`

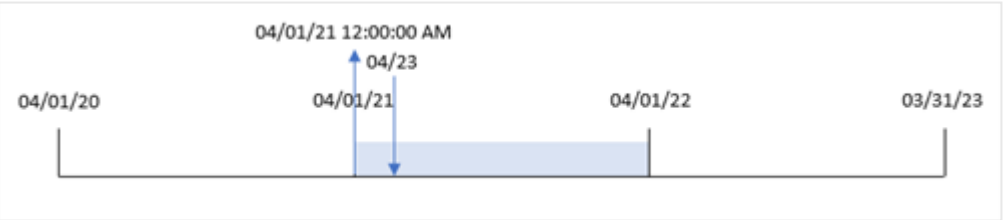
Ergebnistabelle

id	date	year_start	year_start_timestamp
8188	01/13/2020	04/01/2019	4/1/2019 12:00:00 AM
8189	02/26/2020	04/01/2019	4/1/2019 12:00:00 AM
8190	03/27/2020	04/01/2019	4/1/2019 12:00:00 AM
8191	04/16/2020	04/01/2020	4/1/2020 12:00:00 AM
8192	05/21/2020	04/01/2020	4/1/2020 12:00:00 AM

id	date	year_start	year_start_timestamp
8193	08/14/2020	04/01/2020	4/1/2020 12:00:00 AM
8194	10/07/2020	04/01/2020	4/1/2020 12:00:00 AM
8195	12/05/2020	04/01/2020	4/1/2020 12:00:00 AM
8196	01/22/2021	04/01/2020	4/1/2020 12:00:00 AM
8197	02/03/2021	04/01/2020	4/1/2020 12:00:00 AM
8198	03/17/2021	04/01/2020	4/1/2020 12:00:00 AM
8199	04/23/2021	04/01/2021	4/1/2021 12:00:00 AM
8200	05/04/2021	04/01/2021	4/1/2021 12:00:00 AM
8201	06/30/2021	04/01/2021	4/1/2021 12:00:00 AM
8202	07/26/2021	04/01/2021	4/1/2021 12:00:00 AM
8203	12/27/2021	04/01/2021	4/1/2021 12:00:00 AM
8204	06/06/2022	04/01/2022	4/1/2022 12:00:00 AM
8205	07/18/2022	04/01/2022	4/1/2022 12:00:00 AM
8206	11/14/2022	04/01/2022	4/1/2022 12:00:00 AM
8207	12/12/2022	04/01/2022	4/1/2022 12:00:00 AM

Da in diesem Fall das Argument `first_month_of_year` von 4 in der Funktion `yearstart()` verwendet wird, legt sie den ersten Tag des Jahres auf den 1. April und den letzten Tag des Jahres auf den 31. März fest.

Diagramm der Funktion `yearstart()` mit April als erstem Monat



Transaktion 8199 fand am 23. April 2021 statt. Da die Funktion `yearstart()` den Beginn des Jahres auf den 1. April festlegt, gibt sie den Wert „`year_start`“ für die Transaktion zurück.

Beispiel 4 – Diagrammobjektbeispiel

Ladeskript und Diagrammformel

Übersicht

Es werden derselbe Datensatz und dasselbe Szenario wie im ersten Beispiel verwendet.

In diesem Beispiel wird jedoch der unveränderte Datensatz in die Anwendung geladen. Die Berechnung, die den Startdatumszeitstempel für das Jahr zurückgibt, in dem eine Transaktion stattfand, wird als Kennzahl in einem Diagrammobjekt der Anwendung erstellt.

Ladeskript

```
Transactions:
Load
*
Inline
[
id,date,amount
8188,01/13/2020,37.23
8189,02/26/2020,17.17
8190,03/27/2020,88.27
8191,04/16/2020,57.42
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,06/06/2022,46.23
8205,07/18/2022,84.21
8206,11/14/2022,96.24
8207,12/12/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `id`
- `date`

Um zu berechnen, in welchem Jahr eine Transaktion stattfand, erstellen Sie die folgenden Kennzahlen:

- `=yearstart(date)`
- `=timestamp(yearstart(date))`

Ergebnistabelle

id	date	=yearstart(date)	=timestamp(yearstart(date))
8188	06/06/2022	01/01/2022	1/1/2022 12:00:00 AM

id	date	=yearstart(date)	=timestamp(yearstart(date))
8189	07/18/2022	01/01/2022	1/1/2022 12:00:00 AM
8190	11/14/2022	01/01/2022	1/1/2022 12:00:00 AM
8191	12/12/2022	01/01/2022	1/1/2022 12:00:00 AM
8192	01/22/2021	01/01/2021	1/1/2021 12:00:00 AM
8193	02/03/2021	01/01/2021	1/1/2021 12:00:00 AM
8194	03/17/2021	01/01/2021	1/1/2021 12:00:00 AM
8195	04/23/2021	01/01/2021	1/1/2021 12:00:00 AM
8196	05/04/2021	01/01/2021	1/1/2021 12:00:00 AM
8197	06/30/2021	01/01/2021	1/1/2021 12:00:00 AM
8198	07/26/2021	01/01/2021	1/1/2021 12:00:00 AM
8199	12/27/2021	01/01/2021	1/1/2021 12:00:00 AM
8200	01/13/2020	01/01/2020	1/1/2020 12:00:00 AM
8201	02/26/2020	01/01/2020	1/1/2020 12:00:00 AM
8202	03/27/2020	01/01/2020	1/1/2020 12:00:00 AM
8203	04/16/2020	01/01/2020	1/1/2020 12:00:00 AM
8204	05/21/2020	01/01/2020	1/1/2020 12:00:00 AM
8205	08/14/2020	01/01/2020	1/1/2020 12:00:00 AM
8206	10/07/2020	01/01/2020	1/1/2020 12:00:00 AM
8207	12/05/2020	01/01/2020	1/1/2020 12:00:00 AM

Die Kennzahl „start_of_year“ wird im Diagrammobjekt erstellt, indem die Funktion `yearstart()` verwendet und das Feld als Argument der Funktion übergeben wird.

Die Funktion `yearstart()` identifiziert zuerst, in welches Jahr der Datumswert fällt, und gibt einen Zeitstempel für die erste Millisekunde dieses Jahres zurück.

Diagramm von Funktion `yearstart()` und Transaktion 8199

Transaktion 8199 fand am 23. April 2021 statt. Die Funktion `yearstart()` gibt die erste Millisekunde dieses Jahres zurück, also den 1. Januar um 12:00:00 AM.

Beispiel 5 – Szenario

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz wird in eine Tabelle namens „Loans“ geladen. Die Tabelle enthält die folgenden Felder:
    - Darlehens-IDs.
    - Den Saldo zu Beginn des Jahres.
    - Die einfache Zinsrate, die für jedes Darlehen pro Jahr berechnet wird.

Der Endbenutzer möchte ein Diagrammobjekt, das nach Darlehens-ID die aktuellen Zinsen anzeigt, die für jedes Darlehen im Jahr bis dato aufgelaufen sind.

Ladeskript

```
Loans:
Load
*
Inline
[
loan_id,start_balance,rate
8188,$10000.00,0.024
8189,$15000.00,0.057
8190,$17500.00,0.024
8191,$21000.00,0.034
8192,$90000.00,0.084
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `loan_id`
- `start_balance`

Erstellen Sie die folgende Kennzahl, um die kumulierten Zinsen zu berechnen:

```
=start_balance*(rate*(today(1)-yearstart(today(1)))/365)
```

Legen Sie das **Zahlenformat** der Kennzahl auf **Währung** fest.

Ergebnistabelle

loan_id	start_balance	=start_balance*(rate*(today(1)-yearstart(today(1)))/365)
8188	$10000.00	$39.73
8189	$15000.00	$339.66
8190	$17500.00	$166.85
8191	$21000.00	$283.64
8192	$90000.00	$3003.29

Die Funktion `yearstart()` verwendet das aktuelle Datum als einziges Argument und gibt das Startdatum des aktuellen Jahres zurück. Die Formel zieht dieses Ergebnis vom aktuellen Datum ab und gibt die Anzahl der Tage zurück, die bisher im Jahr verstrichen sind.

Dieser Wert wird dann mit dem Zinssatz multipliziert und durch 365 geteilt, um den effektiven Zinssatz für den Zeitraum zurückzugeben. Der effektive Zinssatz für den Zeitraum wird dann mit dem Anfangssaldo des Darlehens multipliziert, was die Zinsen ergibt, die bislang in diesem Jahr aufgelaufen sind.

yeartodate

Diese Funktion findet den Eingabe-Zeitstempel innerhalb des Jahres mit dem Datum, an welchem das Skript zuletzt aufgerufen wurde, und gibt True zurück, wenn der Zeitstempel gefunden wurde bzw. False, wenn er nicht gefunden wurde.

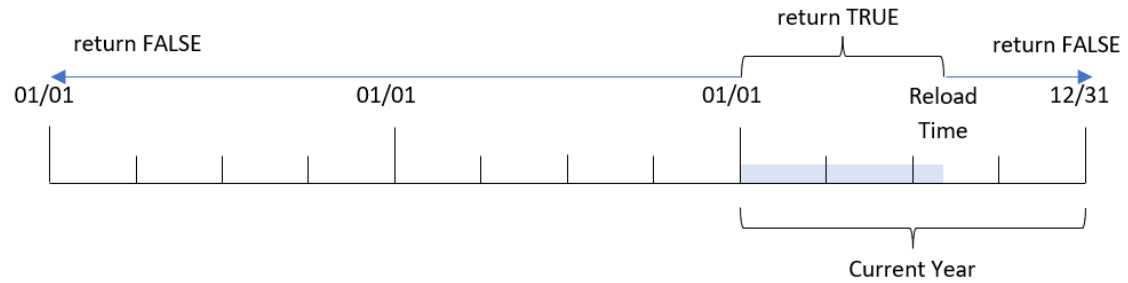
Syntax:

```
YearToDate(timestamp[ , yearoffset [ , firstmonth [ , todaydate] ] ])
```

Rückgabe Datentyp: Boolesch

In Qlik Sense wird der boolesche Wert „wahr“ durch -1 dargestellt, der Wert „falsch“ durch 0.

Beispieldiagramm der Funktion `yeartodate()`



Wird kein weiterer Parameter verwendet, ist das Jahr bis dato das laufende Kalenderjahr, vom 1. Januar bis einschließlich des Tags der letzten Ausführung des Skripts.

In anderen Worten wird die Funktion `yeartodate()`, wenn sie ohne zusätzliche Parameter ausgelöst wird, dazu verwendet, einen Zeitstempel auszuwerten und ein boolesches Ergebnis basierend darauf zurückzugeben, ob das Datum innerhalb des Kalenderjahres bis einschließlich dem Datum liegt, an dem der Ladevorgang stattfand.

Es ist jedoch auch möglich, das Startdatum des Jahres zu überschreiben, indem das Argument `firstmonth` verwendet wird, und es können Vergleiche mit dem davor oder danach liegenden Jahr anhand des Arguments `yearoffset` vorgenommen werden.

Im Fall historischer Datensätze gibt die Funktion `yeartodate()` einen festzulegenden Parameter `todaydate` an, mit dem stattdessen der Zeitstempel mit dem Kalenderjahr bis einschließlich dem Datum verglichen wird, das im Argument `todaydate` angegeben wird.

Argumente

Argument	Beschreibung
timestamp	Der zu beurteilende Zeitstempel, beispielsweise „10/12/2012“.
yearoffset	Durch das Festlegen von **yearoffset**, liefert **yeartodate** für denselben Zeitraum in einem anderen Jahr True. Ein negativer Wert von **yearoffset** verweist auf ein vorheriges Jahr, ein positiver Startwert auf ein Jahr in der Zukunft. Das aktuellste Jahr bis dato wird durch die Festlegung von yearoffset = -1 ausgewählt. Ist nichts definiert, wird 0 angenommen.
firstmonth	Durch Angabe eines Werts für **firstmonth** zwischen 1 und 12 (fehlt die Angabe, wird 1 angenommen) wird der Jahresbeginn auf den ersten Tag eines beliebigen Monats festgelegt. Wenn Sie beispielsweise mit einem Geschäftsjahr arbeiten möchten, das am 1. Mai beginnt, geben Sie **firstmonth** = 5 an. Ein Wert von 1 gibt ein Geschäftsjahr an, das am 1. Januar beginnt, und ein Wert von 12 gibt ein Geschäftsjahr an, das am 1. Dezember beginnt.
todaydate	Durch Angabe einer Zahl für **todaydate** kann das Ende der Zeitperiode festgelegt werden. Fehlt dieser Parameter, wird stattdessen der Zeitstempel der letzten Ausführung des Skripts verwendet.

Verwendung

Die Funktion `yeartodate()` gibt einen booleschen Wert zurück. In der Regel wird dieser Funktionstyp als Bedingung in einem IF-Ausdruck verwendet. Damit wird eine Aggregierung oder Berechnung zurückgegeben, abhängig davon, ob das ausgewertete Datum in das Jahr bis einschließlich zum letzten Ladedatum der Anwendung fällt.

Beispielsweise kann die Funktion „YearToDate()“ verwendet werden, um alle bisher im laufenden Jahr gefertigten Geräte zu identifizieren.

In den folgenden Beispielen wird als letzte Ladezeit 11/18/2011 angenommen.

Funktionsbeispiele

Beispiel	Ergebnis
`yeartodate( '11/18/2010')`	gibt False zurück

Beispiel	Ergebnis
`yeartodate( '02/01/2011')`	gibt True zurück
`yeartodate( '11/18/2011')`	gibt True zurück
`yeartodate( '11/19/2011')`	gibt False zurück
`yeartodate( '11/19/2011', 0, 1, '12/31/2011')`	gibt True zurück
`yeartodate( '11/18/2010', -1)`	gibt True zurück
`yeartodate( '11/18/2011', -1)`	gibt False zurück
`yeartodate( '04/30/2011', 0, 5)`	gibt False zurück
`yeartodate( '05/01/2011', 0, 5)`	gibt True zurück

Regionaleinstellungen

Sofern nicht anders angegeben, verwenden die Beispiele in diesem Thema das folgende Datumsformat: MM/TT/JJJJ. Das Datumsformat wird in der Anweisung `SET DateFormat` in Ihrem Datenladeskript angegeben. Das Standarddatumsformat in Ihrem System kann aufgrund Ihrer regionalen Einstellungen und anderer Faktoren abweichen. Sie können die Formate in den Beispielen unten Ihren Anforderungen entsprechend ändern. Sie können auch die Formate in Ihrem Ladeskript entsprechend den Beispielen ändern.

Die standardmäßigen regionalen Einstellungen in Apps basieren auf den regionalen Systemeinstellungen der Computer oder Server, auf denen Qlik Sense installiert ist. Wenn der Qlik Sense-Server, auf den Sie zugreifen, auf Schweden festgelegt ist, verwendet der Dateneditor die schwedischen regionalen Einstellungen für Datums-, Uhrzeit- und Währungsangaben. Diese Einstellungen im regionalen Format hängen nicht mit der Sprache zusammen, die in der Benutzeroberfläche von Qlik Sense angezeigt wird. Qlik Sense wird in der gleichen Sprache wie der von Ihnen verwendete Browser angezeigt.

Beispiel 1 – einfaches Beispiel

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz, der eine Reihe von Transaktionen zwischen 2020 und 2022 enthält, wird in eine Tabelle namens `Transactions` geladen.
- Das Datumsfeld wird im Format der Systemvariablen `DateFormat` (MM/TT/JJJJ) bereitgestellt.
- Es wird ein Feld `year_to_date` erstellt, das bestimmt, welche Transaktionen im Kalenderjahr bis zum Datum des letzten Ladevorgangs stattfanden.

Das Datum bei Erstellung ist der 26. April 2022.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        yeartodate(date) as year_to_date
        ;
Load
*
Inline
[
id,date,amount
8188,01/10/2020,37.23
8189,02/28/2020,17.17
8190,04/09/2020,88.27
8191,04/16/2020,57.42
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,02/02/2022,46.23
8205,02/26/2022,84.21
8206,03/07/2022,96.24
8207,03/11/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

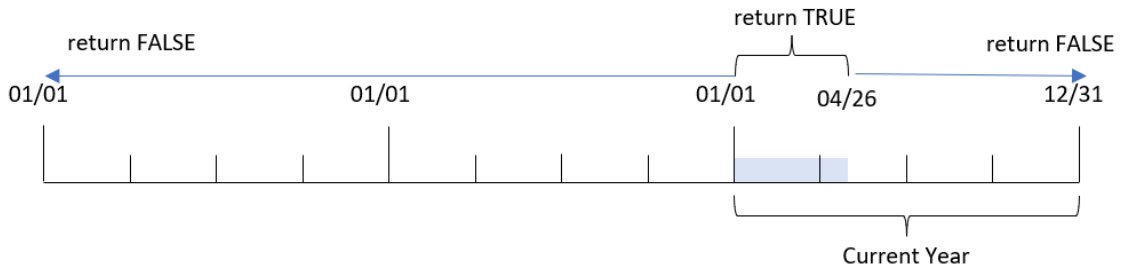
- `date`
- `year_to_date`

Ergebnistabelle

date	year_to_date
01/10/2020	0
02/28/2020	0
04/09/2020	0

date	year_to_date
04/16/2020	0
05/21/2020	0
08/14/2020	0
10/07/2020	0
12/05/2020	0
01/22/2021	0
02/03/2021	0
03/17/2021	0
04/23/2021	0
05/04/2021	0
06/30/2021	0
07/26/2021	0
12/27/2021	0
02/02/2022	-1
02/26/2022	-1
03/07/2022	-1
03/11/2022	-1

Diagramm der Funktion `yeartodate()`, einfaches Beispiel



Das Feld `year_to_date` wird im vorangehenden load-Befehl erstellt, indem die Funktion `yeartodate()` verwendet und das Feld `date` als Argument der Funktion übergeben wird.

Da keine weiteren Parameter an die Funktion übergeben werden, identifiziert die Funktion `yeartodate()` zunächst das Ladedatum und somit die Grenzen für das aktuelle Kalenderjahr (das am 1. Januar beginnt), das ein boolesches Ergebnis von `TRUE` zurückgeben wird.

Daher gibt jede Transaktion, die zwischen dem 1. Januar und dem 26. April, dem Ladedatum, stattfindet, ein boolesches Ergebnis von TRUE zurück Jede Transaktion, die vor Beginn des Jahres 2022 stattfindet, gibt ein boolesches Ergebnis von FALSE zurück.

Beispiel 2 – yearoffset

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Derselbe Datensatz und dasselbe Szenario wie im ersten Beispiel.
- Es wird ein Feld `two_years_prior` erstellt, das bestimmt, welche Transaktionen zwei volle Jahre vor dem Kalenderjahr bis dato stattfanden.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        yeartodate(date,-2) as two_years_prior
        ;
Load
*
Inline
[
id,date,amount
8188,01/10/2020,37.23
8189,02/28/2020,17.17
8190,04/09/2020,88.27
8191,04/16/2020,57.42
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,02/02/2022,46.23
8205,02/26/2022,84.21
8206,03/07/2022,96.24
8207,03/11/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

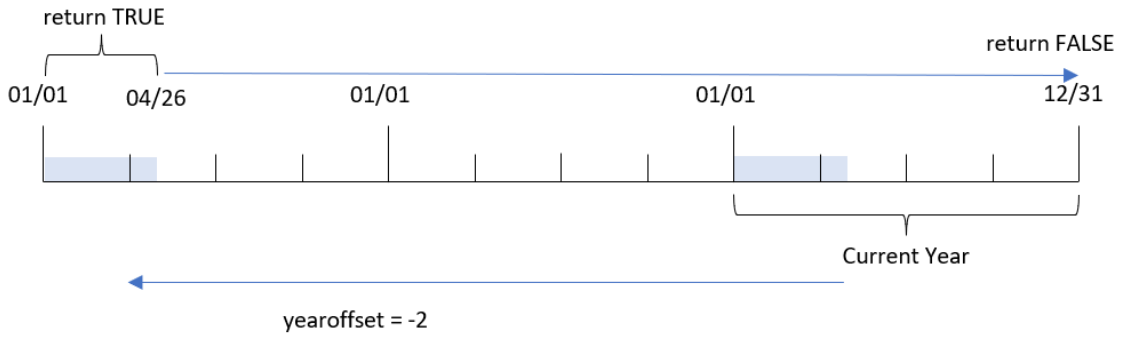
- `date`
- `two_years_prior`

Ergebnistabelle

date	two_years_prior
01/10/2020	-1
02/28/2020	-1
04/09/2020	-1
04/16/2020	-1
05/21/2020	0
08/14/2020	0
10/07/2020	0
12/05/2020	0
01/22/2021	0
02/03/2021	0
03/17/2021	0
04/23/2021	0
05/04/2021	0
06/30/2021	0
07/26/2021	0
12/27/2021	0
02/02/2022	0
02/26/2022	0
03/07/2022	0
03/11/2022	0

Da -2 als das Argument `yearoffset` in der Funktion `yeartodate()` verwendet wird, verschiebt die Funktion die Grenzen des Vergleichs-Kalenderjahressegments um volle zwei Jahre. Anfänglich entspricht das Jahressegment dem 1. Januar bis 26. April 2022. Das Argument `yearoffset` versetzt dieses Segment dann um zwei Jahre zurück. Die Datumsgrenzen fallen somit auf den 1. Januar bis 26. April 2020.

Diagramm der Funktion `yeartodate()`, Beispiel „yearoffset“



Daher gibt jede Transaktion, die zwischen dem 1. Januar und dem 26. April 2020 stattfindet, ein boolesches Ergebnis von TRUE zurück. Alle Transaktionen vor oder nach diesem Segment geben FALSE zurück.

Beispiel 3 – firstmonth

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Derselbe Datensatz und dasselbe Szenario wie im ersten Beispiel.
- Es wird ein Feld `year_to_date` erstellt, das bestimmt, welche Transaktionen im Kalenderjahr bis zum Datum des letzten Ladevorgangs stattfanden.

In diesem Beispiel wird der Start des Geschäftsjahres auf den 1. Juli festgelegt.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
    Load
        *,
        yeartodate(date,0,7) as year_to_date
        ;
Load
*
Inline
[
id,date,amount
8188,01/10/2020,37.23
8189,02/28/2020,17.17
8190,04/09/2020,88.27
8191,04/16/2020,57.42
```

```
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,02/02/2022,46.23
8205,02/26/2022,84.21
8206,03/07/2022,96.24
8207,03/11/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `year_to_date`

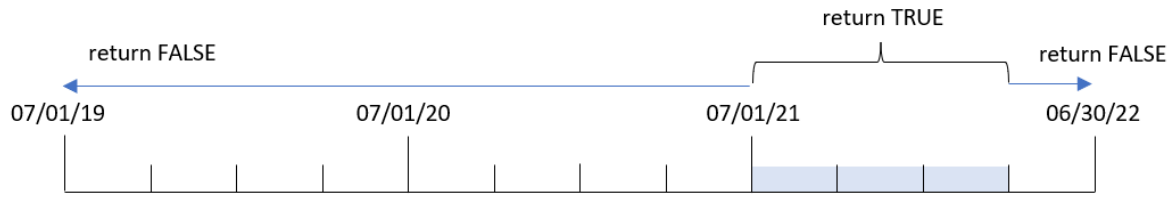
Ergebnistabelle

date	year_to_date
01/10/2020	0
02/28/2020	0
04/09/2020	0
04/16/2020	0
05/21/2020	0
08/14/2020	0
10/07/2020	0
12/05/2020	0
01/22/2021	0
02/03/2021	0
03/17/2021	0
04/23/2021	0
05/04/2021	0
06/30/2021	0

date	year_to_date
07/26/2021	-1
12/27/2021	-1
02/02/2022	-1
02/26/2022	-1
03/07/2022	-1
03/11/2022	-1

Da in diesem Fall das Argument `firstmonth` von 7 in der Funktion `yeartodate()` verwendet wird, legt sie den ersten Tag des Jahres auf den 1. Juli und den letzten Tag des Jahres auf den 30. Juni fest.

Diagramm der Funktion `yeartodate()`, Beispiel „firstmonth“



Daher gibt jede Transaktion, die zwischen dem 1. Juli 2021 und dem 26. April 2022 stattfindet, ein boolesches Ergebnis von TRUE zurück. Jede Transaktion, die vor dem 1. Juli 2021 stattfindet, gibt ein boolesches Ergebnis von FALSE zurück.

Beispiel 4 – todaydate

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Derselbe Datensatz und dasselbe Szenario wie im ersten Beispiel.
- Es wird ein Feld `year_to_date` erstellt, das bestimmt, welche Transaktionen im Kalenderjahr bis zum Datum des letzten Ladevorgangs stattfanden.

In diesem Beispiel müssen wir jedoch alle Transaktionen identifizieren, die im Kalenderjahr bis einschließlich 1. März 2022 stattgefunden haben.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';
```

```
Transactions:
    Load
        *,
        yeartodate(date, 0, 1, '03/01/2022') as year_to_date
;
Load
*
Inline
[
id,date,amount
8188,01/10/2020,37.23
8189,02/28/2020,17.17
8190,04/09/2020,88.27
8191,04/16/2020,57.42
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,02/02/2022,46.23
8205,02/26/2022,84.21
8206,03/07/2022,96.24
8207,03/11/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `date`
- `year_to_date`

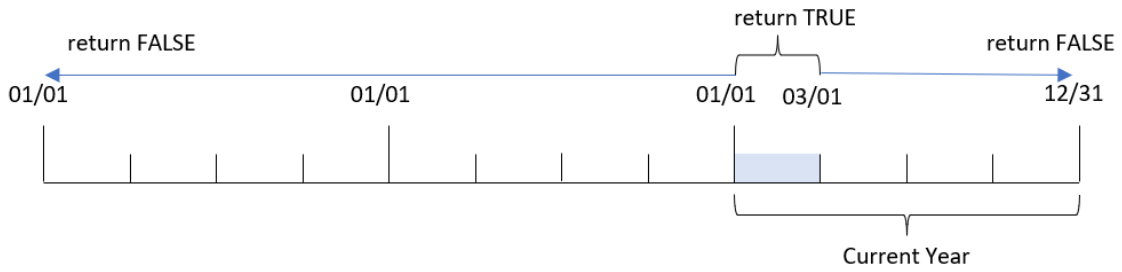
Ergebnistabelle

date	year_to_date
01/10/2020	0
02/28/2020	0
04/09/2020	0
04/16/2020	0
05/21/2020	0

date	year_to_date
08/14/2020	0
10/07/2020	0
12/05/2020	0
01/22/2021	0
02/03/2021	0
03/17/2021	0
04/23/2021	0
05/04/2021	0
06/30/2021	0
07/26/2021	0
12/27/2021	0
02/02/2022	-1
02/26/2022	-1
03/07/2022	0
03/11/2022	0

Da in diesem Fall das Argument `todaydate` von 03/01/2022 in der Funktion `yeartodate()` verwendet wird, wird die Endgrenze des Vergleichsjahressegments auf den 1. März 2022 festgelegt. Der Parameter `firstmonth` (zwischen 1 und 12) muss unbedingt angegeben werden; andernfalls gibt die Funktion Nullwerte zurück.

Diagramm der Funktion `yeartodate()`, Beispiel mit dem Argument „todaydate“



Daher gibt der Parameter `todaydate` für jede Transaktion, die zwischen dem 1. Januar 2022 und dem 1. März 2022 stattfindet, ein boolesches Ergebnis von TRUE zurück. Jede Transaktion, die vor dem 1. Januar 2022 oder nach dem 1. März 2022 stattfindet, gibt ein boolesches Ergebnis von FALSE zurück.

Beispiel 5 – Diagrammobjektbeispiel

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript verwendet den gleichen Datensatz und das gleiche Szenario wie das erste Beispiel.

In diesem Beispiel wird jedoch der unveränderte Datensatz in die Anwendung geladen. Die Berechnung, die bestimmt, ob Transaktionen im Kalenderjahr bis einschließlich zum Datum des letzten Ladevorgangs stattfanden, wird als Kennzahl in einem Diagrammobjekt der Anwendung erstellt.

Ladeskript

```
Transactions:
Load
*
Inline
[
id,date,amount
8188,01/10/2020,37.23
8189,02/28/2020,17.17
8190,04/09/2020,88.27
8191,04/16/2020,57.42
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,02/02/2022,46.23
8205,02/26/2022,84.21
8206,03/07/2022,96.24
8207,03/11/2022,67.67
];
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie dieses Feld als Dimension hinzu: `date`.

Fügen Sie die folgende Kennzahl hinzu:

```
=yeartodate(date)
```

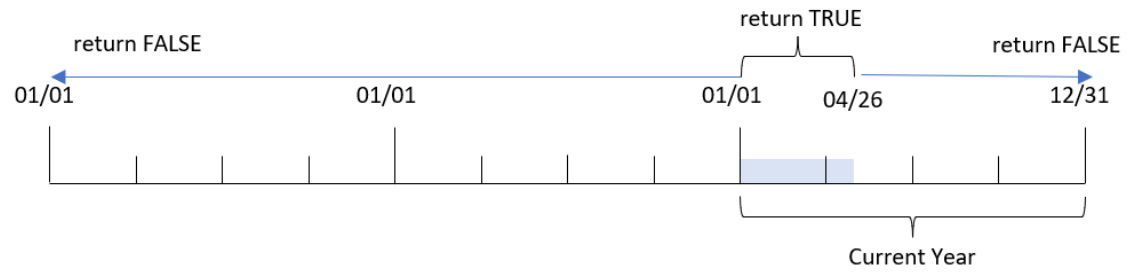
Ergebnistabelle

date	=yeartodate(date)
01/10/2020	0
02/28/2020	0
04/09/2020	0
04/16/2020	0
05/21/2020	0
08/14/2020	0
10/07/2020	0
12/05/2020	0
01/22/2021	0
02/03/2021	0
03/17/2021	0
04/23/2021	0
05/04/2021	0
06/30/2021	0
07/26/2021	0
12/27/2021	0
02/02/2022	-1
02/26/2022	-1
03/07/2022	-1
03/11/2022	-1

Die Kennzahl `year_to_date` wird im Diagrammobjekt erstellt, indem die Funktion `yeartodate()` verwendet und das Feld `date` als Argument der Funktion übergeben wird.

Da keine weiteren Parameter an die Funktion übergeben werden, identifiziert die Funktion `yeartodate()` zunächst das Ladedatum und somit die Grenzen für das aktuelle Kalenderjahr (das am 1. Januar beginnt), das ein boolesches Ergebnis von TRUE zurückgeben wird.

Diagramm der Funktion `yeartodate()`, Diagrammobjektbeispiel



Jede Transaktion, die zwischen dem 1. Januar und dem 26. April, dem Ladedatum, stattfindet, gibt ein boolesches Ergebnis von TRUE zurück. Jede Transaktion, die vor Beginn des Jahres 2022 stattfindet, gibt ein boolesches Ergebnis von FALSE zurück.

Beispiel 6 – Szenario

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Ein Datensatz, der eine Reihe von Transaktionen zwischen 2020 und 2022 enthält, wird in eine Tabelle namens `Transactions` geladen.
- Datumsfeld, das im Format der Systemvariablen `DateFormat` (MM/TT/JJJJ) bereitgestellt wird

Der Endbenutzer möchte ein KPI-Objekt, das den Gesamtumsatz für den entsprechenden Zeitraum im Jahr 2021 wie im aktuellen Jahr bis dato als letzte Ladezeit darstellt.

Das Datum bei Erstellung ist der 16. Juni 2022.

Ladeskript

```
SET DateFormat='MM/DD/YYYY';

Transactions:
Load
*
Inline
[
id,date,amount
8188,01/10/2020,37.23
8189,02/28/2020,17.17
8190,04/09/2020,88.27
8191,04/16/2020,57.42
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
```

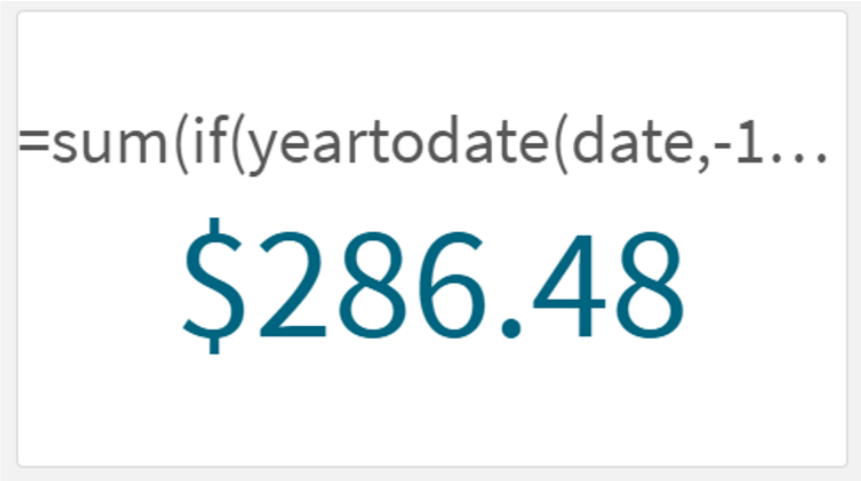
```
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,02/02/2022,46.23
8205,02/26/2022,84.21
8206,03/07/2022,96.24
8207,03/11/2022,67.67
];
```

Ergebnisse

Gehen Sie folgendermaßen vor:

1. Erstellen Sie ein KPI-Objekt.
2. Erstellen Sie die folgende Aggregierungskennzahl, um den Gesamtumsatz zu berechnen: `=sum(if(yeartodate(date,-1),amount,0))`
3. Legen Sie das **Zahlenformat** der Kennzahl auf **Währung** fest.

KPI-Diagramm `yeartodate()` für 2021



Die Funktion `yeartodate()` gibt einen booleschen Wert zurück, wenn sie die Datumswerte für jede Transaktions-ID auswertet. Da der Ladevorgang am 16. Juni 2022 stattfand, segmentiert die Funktion `yeartodate` den Jahreszeitraum auf den Zeitraum zwischen dem 01/01/2022 und dem 06/16/2022. Da aber ein Wert für `period_no` von -1 in der Funktion verwendet wurde, werden diese Grenzen dann auf das vorherige Jahr verschoben. Somit gibt die Funktion `yeartodate()` für jede Transaktion, die zwischen dem 01/01/2021 und dem 06/16/2021 stattgefunden hat, einen booleschen Wert von `TRUE` zurück und summiert den Betrag.

8.8 Exponential- und Logarithmusfunktionen

In diesem Abschnitt werden die Funktionen in Bezug auf Exponential- und Logarithmusberechnungen beschrieben. Alle Funktionen können sowohl im Datenladeskript als auch in den Diagrammformeln verwendet werden.

In den nachfolgend beschriebenen Funktionen stehen die Parameter **x** und **y** für reelle Zahlen.

exp
Die natürliche Exponentialfunktion, e^x, die den natürlichen Logarithmus **e** als Basis nutzt. Das Ergebnis ist eine positive Zahl.

```
exp(x )
```

Beispiele und Ergebnisse:

`exp(3)` liefert 20.085.

log
Natürlicher Logarithmus von **x**. Diese Funktion ist nur für **x**> 0 definiert. Das Ergebnis ist eine Zahl.

```
log(x )
```

Beispiele und Ergebnisse:

`log(3)` liefert 1,0986

log10
Der Zehner-Logarithmus von **x** (Logarithmus mit Basis 10). Diese Funktion ist nur für **x**> 0 definiert. Das Ergebnis ist eine Zahl.

```
log10(x )
```

Beispiele und Ergebnisse:

`log10(3)` liefert 0,4771

pow
Liefert **x** hoch **y**. Das Ergebnis ist eine Zahl.

```
pow(x,y )
```

Beispiele und Ergebnisse:

`pow(3, 3)` liefert 27

sqr
x zum Quadrat (**x** hoch 2). Das Ergebnis ist eine Zahl.

```
sqr (x )
```

Beispiele und Ergebnisse:

`sqr(3)` liefert 9

sqrt
Quadratwurzel von **x**. Diese Funktion ist nur für **x** >= 0 definiert. Das Ergebnis ist eine positive Zahl.

```
sqrt(x )
```

Beispiele und Ergebnisse:

`sqrt(3)` liefert 1,732

8.9 Feldfunktionen

Diese Funktionen können ausschließlich in den Formeln von Diagrammen verwendet werden.

Feldfunktionen liefern entweder ganze Zahlen oder Strings, die verschiedene Aspekte der Feldauswahlen kennzeichnen.

Counter-Funktionen

GetAlternativeCount
GetAlternativeCount() liefert die Anzahl der alternativen (hellgrauen) Werte im identifizierten Feld.

```
GetAlternativeCount - Diagrammfunktion (field_name)
```

GetExcludedCount
GetExcludedCount() liefert die Anzahl der ausgeschlossenen distinkten Werte im identifizierten Feld. Zu den ausgeschlossenen Werten gehören alternative Felder (hellgrau), ausgeschlossene Felder (dunkelgrau) und ausgewählte ausgeschlossene Felder (dunkelgrau mit Häkchen).

```
GetExcludedCount - Diagrammfunktion (page 1220)(field_name)
```

GetNotSelectedCount
Diese Diagrammfunktion liefert die Zahl nicht ausgewählter Werte des Feldes **fieldname**. Diese Funktion liefert nur bei Feldern im Und-Modus Ergebnisse.

```
GetNotSelectedCount - Diagrammfunktion(fieldname [, includeexcluded=false])
```

GetPossibleCount
GetPossibleCount() liefert die Anzahl der möglichen Werte im identifizierten Feld. Wenn das identifizierte Feld Auswahlen enthält, werden die ausgewählten (grünen) Felder gezählt. Ansonsten werden verknüpfte (weiße) Werte gezählt.

```
GetPossibleCount - Diagrammfunktion(field_name)
```

GetSelectedCount

GetSelectedCount() liefert die Anzahl der ausgewählten (grünen) Werte in einem Feld.

```
GetSelectedCount - Diagrammfunktion (field_name [, include_excluded])
```

Feld- und Auswahlfunktionen

GetCurrentSelections

GetCurrentSelections() gibt eine Liste der aktuellen Auswahlen in der App zurück. Wenn die Auswahlen stattdessen anhand einer Suchzeichenfolge in einem Suchfeld getroffen werden, gibt **GetCurrentSelections()** die Suchzeichenfolge zurück.

```
GetCurrentSelections - Diagrammfunktion([record_sep [,tag_sep [,value_sep
[,max_values]]]])
```

GetFieldSelections

GetFieldSelections() liefert einen **String** mit der aktuellen Auswahl in einem Feld.

```
GetFieldSelections - Diagrammfunktion ( field_name [, value_sep [, max_
values]])
```

GetObjectDimension

GetObjectDimension() gibt den Namen der Dimension zurück. **Index** ist eine optionale Ganzzahl, die die Dimension angibt, die zurückgegeben werden soll.

```
GetObjectDimension - Diagrammfunktion ([index])
```

GetObjectField

GetObjectField() gibt den Namen der Dimension zurück. **Index** ist eine optionale Ganzzahl, die die Dimension angibt, die zurückgegeben werden sollte.

```
GetObjectField - Diagrammfunktion ([index])
```

GetObjectMeasure

GetObjectMeasure() gibt den Namen der Kennzahl zurück. **Index** ist eine optionale Ganzzahl, die die Kennzahl angibt, die zurückgegeben werden sollte.

```
GetObjectMeasure - Diagrammfunktion ([index])
```

GetAlternativeCount - Diagrammfunktion

GetAlternativeCount() liefert die Anzahl der alternativen (hellgrauen) Werte im identifizierten Feld.

Syntax:

```
GetAlternativeCount (field_name)
```

Rückgabe Datentyp: ganze Zahl

Argumente:

Argumente

Argument	Beschreibung
field_name	Das Feld mit dem Datenbereich, der angegeben werden soll.

Beispiele und Ergebnisse:

Im folgenden Beispiel wird das in ein Filterfenster geladene Feld **First name** verwendet.

Beispiele und Ergebnisse

Beispiele	Ergebnisse
Vorgabe: **John** ist in **First name** ausgewählt. `GetAlternativeCount ([First name])`	4, da 4 Werte in **First name** eindeutig und ausgeschlossen (grau) sind.
Vorgabe: **John** und **Peter** sind ausgewählt. `GetAlternativeCount ([First name])`	3, da 3 Werte in **First name** eindeutig und ausgeschlossen (grau) sind.
Vorgabe: Keine Werte sind in **First name** ausgewählt. `GetAlternativeCount ([First name])`	0, da keine Auswahlen vorhanden sind.

Im Beispiel verwendete Daten:

```
Names:
LOAD * inline [
First name|Last name|Initials|Has cellphone
John|Anderson|JA|Yes
Sue|Brown|SB|Yes
Mark|Carr|MC|No
Peter|Devonshire|PD|No
Jane|Elliot|JE|Yes
Peter|Franc|PF|Yes ] (delimiter is '|');
```

GetCurrentSelections - Diagrammfunktion

GetCurrentSelections() gibt eine Liste der aktuellen Auswahlen in der App zurück. Wenn die Auswahlen stattdessen anhand einer Suchzeichenfolge in einem Suchfeld getroffen werden, gibt **GetCurrentSelections()** die Suchzeichenfolge zurück.

Wenn Optionen verwendet werden sollen, müssen Sie record_sep definieren. Definieren Sie eine neue Zeile, indem Sie **record_sep** als **chr(13)&chr(10)** angeben.

Wenn alle bis auf zwei oder alle bis auf einen Wert ausgewählt sind, wird das Format 'NOT x,y' beziehungsweise 'NOT y' verwendet. Wenn Sie alle Werte auswählen und die Anzahl aller Werte größer als max_values ist, wird der Text ALL ausgegeben.

Syntax:

```
GetCurrentSelections ([record_sep [, tag_sep [, value_sep [, max_values [, state_name]]]]])
```

Rückgabe Datentyp: String

Argumente:

Argumente

Argumente	Beschreibung
record_sep	Trennzeichen zwischen Felddatensätzen. Standard ist <CR><LF>, d. h. eine neue Zeile.
tag_sep	Trennzeichen zwischen dem Feldnamen und den Feldwerten. Standard ist ':'.
value_sep	Das Trennzeichen zwischen Feldwerten. Standard ist ','.
max_values	Die maximale Zahl der einzeln aufgeführten Feldwerte. Wird eine größere Anzahl an Werten ausgewählt, wird das Format 'x von y Werten' verwendet. Standard ist 6.
state_name	Der Name eines alternativen Status, der für diese Visualisierung ausgewählt wurde. Beim Argument **state_name** werden nur die Auswahlen für den angegebenen Statusnamen berücksichtigt.

Beispiele und Ergebnisse:

Im folgenden Beispiel werden zwei Felder verwendet, die in verschiedene Filterfenster geladen wurden: eins für **First name** und eins für **Initials**.

Beispiele und Ergebnisse

Beispiele	Ergebnisse
Vorgabe: **John** ist in **First name** ausgewählt. `GetCurrentSelections ()`	'First name: John'
Vorgabe: **John** und **Peter** sind in **First name** ausgewählt. `GetCurrentSelections ()`	'First name: John, Peter'
Vorgabe: **John** und **Peter** sind in **First name** ausgewählt und **JA** ist in **Initials** ausgewählt. `GetCurrentSelections ()`	'First name: John, Peter Initials: JA'
Vorgabe: **John** ist in **First name** ausgewählt und **JA** ist in **Initials** ausgewählt. `GetCurrentSelections ( chr(13)&chr(10) , ' = ' )`	'First name = John Initials = JA'

Beispiele	Ergebnisse
Vorgabe: Sie haben alle Namen außer Sue in **First name** und keine Auswahlen in **Initials** vorgenommen. `GetCurrentSelections (chr(13)&chr(10), '=', ',' ,3)`	'First name=NOT Sue'

Im Beispiel verwendete Daten:

```
Names:
LOAD * inline [
First name|Last name|Initials|Has cellphone
John|Anderson|JA|Yes
Sue|Brown|SB|Yes
Mark|Carr|MC|No
Peter|Devonshire|PD|No
Jane|Elliot|JE|Yes
Peter|Franc|PF|Yes ] (delimiter is '|');
```

GetExcludedCount - Diagrammfunktion

GetExcludedCount() liefert die Anzahl der ausgeschlossenen distinkten Werte im identifizierten Feld. Zu den ausgeschlossenen Werten gehören alternative Felder (hellgrau), ausgeschlossene Felder (dunkelgrau) und ausgewählte ausgeschlossene Felder (dunkelgrau mit Häkchen).

Syntax:

```
GetExcludedCount (field_name)
```

Rückgabe Datentyp: String

Argumente:

Argumente

Argumente	Beschreibung
field_name	Das Feld mit dem Datenbereich, der angegeben werden soll.

Beispiele und Ergebnisse:

Im folgenden Beispiel werden drei Felder verwendet, die in verschiedene Filterfenster geladen wurden: eins für **First name**, eins für **Last name** und eins für **Initials**.

Beispiele und Ergebnisse

Beispiele	Ergebnisse
Wenn in **First name** keine Werte ausgewählt sind.	`GetExcludedCount (Initials) = 0` Es sind keine Auswahlen vorhanden.

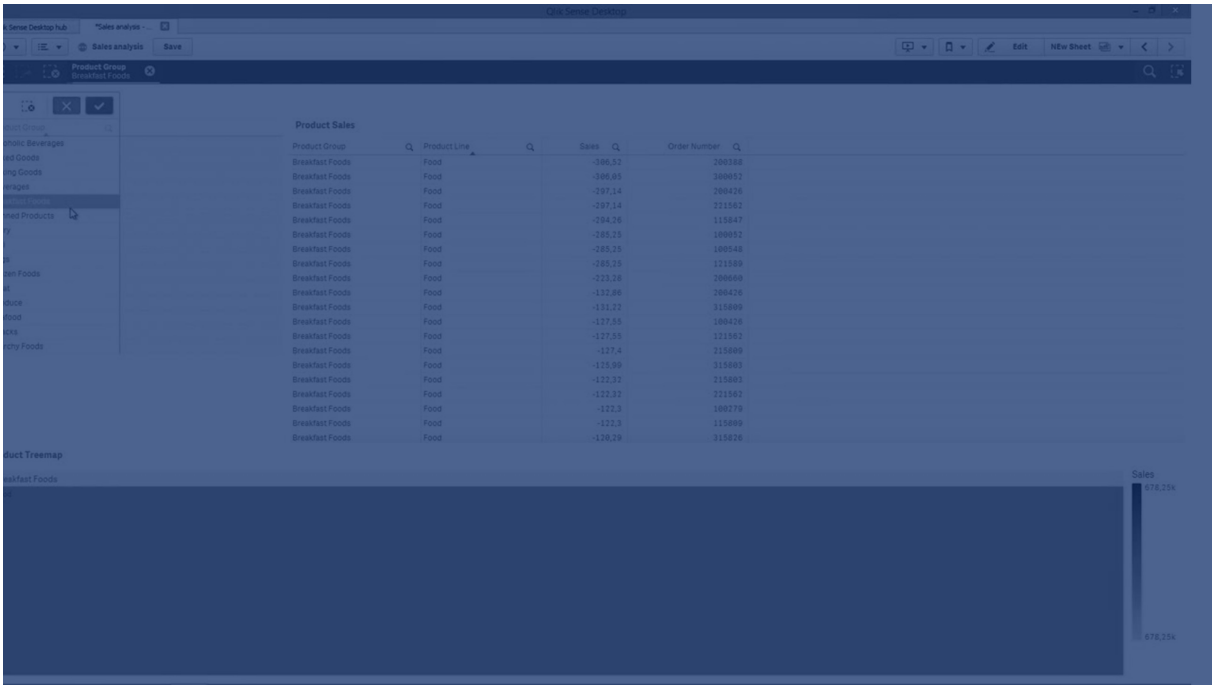
Beispiele	Ergebnisse
Wenn **John** in **First name** ausgewählt ist.	`GetExcludedCount (Initials) = 5` Es sind 5 ausgeschlossene Werte in **Initials** mit dunkelgrauer Farbe vorhanden. Die sechste Zelle (JA) ist weiß, da sie der Auswahl John in **First name** zugeordnet ist.
Wenn **John** und **Peter** ausgewählt sind.	`GetExcludedCount (Initials) = 3` In **Initials** ist John 1 Wert zugeordnet und Peter ist 2 Werten zugeordnet.
Wenn **John** und **Peter** in **First name** ausgewählt sind und dann **Franc** in **Last name** ausgewählt wird.	`GetExcludedCount ([First name]) = 4` In **First name** sind 4 ausgeschlossene Werte mit dunkelgrauer Farbe vorhanden. **GetExcludedCount()** wird für Felder mit ausgeschlossenen Werten ausgewertet, einschließlichen alternativer und ausgewählter ausgeschlossener Felder.
Wenn **John** und **Peter** in **First name** ausgewählt sind und dann **Franc** und **Anderson** in **Last name** ausgewählt werden.	`GetExcludedCount (Initials) = 4` Es sind 4 ausgeschlossene Werte in **Initials** mit dunkelgrauer Farbe vorhanden. Die anderen beiden Zellen (JA und PF) sind weiß, da sie den Auswahlen John und Peter in **First name** zugeordnet sind.
Wenn **John** und **Peter** in **First name** ausgewählt sind und dann **Franc** und **Anderson** in **Last name** ausgewählt werden.	`GetExcludedCount ([Last name]) = 4` Es sind 4 ausgeschlossene Werte in **Initials** vorhanden. Devonshire hat eine hellgraue Farbe, während Brown, Carr und Elliot eine dunkelgraue Farbe haben.

Im Beispiel verwendete Daten:

```
Names:
LOAD * inline [
First name|Last name|Initials|Has cellphone
John|Anderson|JA|Yes
Sue|Brown|SB|Yes
Mark|Carr|MC|No
Peter|Devonshire|PD|No
Jane|Elliot|JE|Yes
Peter|Franc|PF|Yes ] (delimiter is '|');
```

GetFieldSelections - Diagrammfunktion

GetFieldSelections() liefert einen **String** mit der aktuellen Auswahl in einem Feld.



Wenn alle bis auf zwei oder alle bis auf einen Wert ausgewählt sind, wird das Format 'NOT x,y' beziehungsweise 'NOT y' verwendet. Wenn Sie alle Werte auswählen und die Anzahl aller Werte größer als max_values ist, wird der Text ALL ausgegeben.

Syntax:

```
GetFieldSelections ( field_name [, value_sep [, max_values [, state_name]]])
```

Rückgabe Datentyp: String

Rückgabe-String-Formate

Format	Beschreibung
'a, b, c'	Wenn die Anzahl der ausgewählten Werte max_values oder weniger beträgt, ist der zurückgegebene String eine Liste der ausgewählten Werte. Die Werte werden mit value_sep als Trennzeichen getrennt.
'NOT a, b, c'	Wenn die Anzahl der nicht ausgewählten Werte max_values oder weniger beträgt, ist der zurückgegebene String eine Liste der nicht ausgewählten Werte mit vorangestelltem NOT. Die Werte werden mit value_sep als Trennzeichen getrennt.
'x of y'	x = Anzahl der ausgewählten Werte y = Gesamtzahl der Werte Dies wird zurückgegeben, wenn max_values < x < (y - max_values).
'ALL'	Wird zurückgegeben, wenn alle Werte ausgewählt sind.
'-'	Wird zurückgegeben, wenn kein Wert ausgewählt ist.
<search string>	Wenn Sie anhand einer Suche ausgewählt haben, wird der Such-String zurückgegeben.

Argumente:

Argumente

Argumente	Beschreibung
field_name	Das Feld mit dem Datenbereich, der angegeben werden soll.
value_sep	Das Trennzeichen zwischen Feldwerten. Standard ist ','.
max_values	Die maximale Zahl der einzeln aufgeführten Feldwerte. Wird eine größere Anzahl an Werten ausgewählt, wird das Format 'x von y Werten' verwendet. Standard ist 6.
state_name	Der Name eines alternativen Status, der für diese Visualisierung ausgewählt wurde. Beim Argument **state_name** werden nur die Auswahlen für den angegebenen Statusnamen berücksichtigt.

Beispiele und Ergebnisse:

Im folgenden Beispiel wird das in ein Filterfenster geladene Feld **First name** verwendet.

Beispiele und Ergebnisse

Beispiele	Ergebnisse
Vorgabe: **John** ist in **First name** ausgewählt. `GetFieldSelections ([First name])`	'John'
Vorgabe: **John** und **Peter** sind ausgewählt. `GetFieldSelections ([First name])`	'John,Peter'
Vorgabe: **John** und **Peter** sind ausgewählt. `GetFieldSelections ([First name],'; ')`	'John; Peter'
Vorgabe: **John**, **Sue**, **Mark** sind in **First name** ausgewählt. `GetFieldSelections ([First name],';',2)`	'NOT Jane;Peter', weil der Wert 2 als Wert des Arguments max_values angegeben ist. Anderenfalls wäre das Ergebnis Folgendes gewesen: John; Sue; Mark.

Im Beispiel verwendete Daten:

```
Names:
LOAD * inline [
First name|Last name|Initials|Has cellphone
John|Anderson|JA|Yes
```

```
Sue|Brown|SB|Yes
Mark|Carr|MC|No
Peter|Devonshire|PD|No
Jane|Elliot|JE|Yes
Peter|Franc|PF|Yes ] (delimiter is '|');
```

GetNotSelectedCount - Diagrammfunktion

Diese Diagrammfunktion liefert die Zahl nicht ausgewählter Werte des Feldes **fieldname**. Diese Funktion liefert nur bei Feldern im Und-Modus Ergebnisse.

Syntax:

```
GetNotSelectedCount(fieldname [, includeexcluded=false])
```

Argumente:

Argumente

Argument	Beschreibung
fieldname	Der Name des auszuwertenden Felds.
includeexcluded	Ergibt **includeexcluded** True, werden auch ausgewählte Werte mitgezählt, die durch eine Auswahl in anderen Feldern ausgeschlossen sind.

Beispiele:

```
GetNotSelectedCount( Country )
```

```
GetNotSelectedCount( Country, true )
```

GetObjectDimension - Diagrammfunktion

GetObjectDimension() gibt den Namen der Dimension zurück. **Index** ist eine optionale Ganzzahl, die die Dimension angibt, die zurückgegeben werden soll.

Sie können diese Funktion in einem Diagramm an folgenden Stellen nicht verwenden: Titel, Untertitel, Fußzeile, Positionslinienformel und min/max-Formel.

Sie können den Namen einer Dimension oder Kennzahl nicht in einem anderen Objekt unter Verwendung der Object ID referenzieren.

Syntax:

```
GetObjectDimension ([index])
```

Beispiel:

```
GetObjectDimension(1)
```

Beispiel: Diagrammformel

Qlik Sense Tabelle mit Beispielen der GetObjectDimension-Funktion in einer Diagrammformel

transaction_date	customer_id	transaction_quantity	=GetObjectDimension ()	=GetObjectDimension (0)	=GetObjectDimension (1)
2018/08/30	049681	13	transaction_date	transaction_date	customer_id
2018/08/30	203521	6	transaction_date	transaction_date	customer_id
2018/08/30	203521	21	transaction_date	transaction_date	customer_id

Wenn Sie den Namen einer Kennzahl zurückgeben möchten, verwenden Sie stattdessen die Funktion **GetObjectMeasure**.

GetObjectField - Diagrammfunktion

GetObjectField() gibt den Namen der Dimension zurück. **Index** ist eine optionale Ganzzahl, die die Dimension angibt, die zurückgegeben werden sollte.

Sie können diese Funktion in einem Diagramm an folgenden Stellen nicht verwenden: Titel, Untertitel, Fußzeile, Positionslinienformel und min/max-Formel.

Sie können den Namen einer Dimension oder Kennzahl nicht in einem anderen Objekt unter Verwendung der Object ID referenzieren.

Syntax:

```
GetObjectField ([index])
```

Beispiel:

```
GetObjectField(1)
```

Beispiel: Diagrammformel

Qlik Sense Tabelle mit Beispielen der GetObjectField-Funktion in einer Diagrammformel.

transaction_date	customer_id	transaction_quantity	=GetObjectField ()	=GetObjectField (0)	=GetObjectField (1)
2018/08/30	049681	13	transaction_date	transaction_date	customer_id
2018/08/30	203521	6	transaction_date	transaction_date	customer_id
2018/08/30	203521	21	transaction_date	transaction_date	customer_id

Wenn Sie den Namen einer Kennzahl zurückgeben möchten, verwenden Sie stattdessen die Funktion **GetObjectMeasure**.

GetObjectMeasure - Diagrammfunktion

GetObjectMeasure() gibt den Namen der Kennzahl zurück. **Index** ist eine optionale Ganzzahl, die die Kennzahl angibt, die zurückgegeben werden sollte.

Sie können diese Funktion in einem Diagramm an folgenden Stellen nicht verwenden: Titel, Untertitel, Fußzeile, Positionslinienformel und min/max-Formel.

Sie können den Namen einer Dimension oder Kennzahl nicht in einem anderen Objekt unter Verwendung der Object ID referenzieren.

Syntax:

```
GetObjectMeasure ([index])
```

Beispiel:

```
GetObjectMeasure(1)
```

Beispiel: Diagrammformel

Qlik Sense Tabelle mit Beispielen der GetObjectMeasure-Funktion in einer Diagrammformel

customer_id	sum (transaction_quantity)	Avg (transaction_quantity)	=GetObjectMeasure ()	=GetObjectMeasure(0)	=GetObjectMeasure(1)
49681	13	13	sum(transaction_quantity)	sum(transaction_quantity)	Avg(transaction_quantity)
203521	27	13.5	sum(transaction_quantity)	sum(transaction_quantity)	Avg(transaction_quantity)

Wenn Sie den Namen einer Dimension zurückgeben möchten, verwenden Sie stattdessen die Funktion **GetObjectField**.

GetPossibleCount - Diagrammfunktion

GetPossibleCount() liefert die Anzahl der möglichen Werte im identifizierten Feld. Wenn das identifizierte Feld Auswahlen enthält, werden die ausgewählten (grünen) Felder gezählt. Ansonsten werden verknüpfte (weiße) Werte gezählt.

GetPossibleCount() liefert für Felder mit Auswahlen die Anzahl der ausgewählten (grünen) Felder.

Rückgabe Datentyp: ganze Zahl

Syntax:

```
GetPossibleCount (field_name)
```

Argumente:

Argumente

Argumente	Beschreibung
field_name	Das Feld mit dem Datenbereich, der angegeben werden soll.

Beispiele und Ergebnisse:

Im folgenden Beispiel werden zwei Felder verwendet, die in verschiedene Filterfenster geladen wurden: eins für **First name** und eins für **Initials**.

Beispiele und Ergebnisse

Beispiele	Ergebnisse
Vorgabe: **John** ist in **First name** ausgewählt. `GetPossibleCount ([Initials])`	1, da 1 Wert in Initials mit der Auswahl verknüpft ist: **John** in **First name**.
Vorgabe: **John** ist in **First name** ausgewählt. `GetPossibleCount ([First name])`	1, da 1 Auswahl vorhanden ist: **John** in **First name**.
Vorgabe: **Peter** ist in **First name** ausgewählt. `GetPossibleCount ([Initials])`	2, da Peter mit 2 Werten in **Initials** verknüpft ist.
Vorgabe: Keine Werte sind in **First name** ausgewählt. `GetPossibleCount ([First name])`	5, da keine Auswahlen, aber 5 eindeutige Werte in **First name** vorhanden sind.
Vorgabe: Keine Werte sind in **First name** ausgewählt. `GetPossibleCount ([Initials])`	6, da keine Auswahlen, aber 6 eindeutige Werte in **Initials** vorhanden sind.

Im Beispiel verwendete Daten:

```
Names:
LOAD * inline [
First name|Last name|Initials|Has cellphone
John|Anderson|JA|Yes
Sue|Brown|SB|Yes
Mark|Carr|MC|No
Peter|Devonshire|PD|No
Jane|Elliot|JE|Yes
Peter|Franc|PF|Yes ] (delimiter is '|');
```

GetSelectedCount - Diagrammfunktion

GetSelectedCount() liefert die Anzahl der ausgewählten (grünen) Werte in einem Feld.

Syntax:

```
GetSelectedCount (field_name [, include_excluded [, state_name]])
```

Rückgabe Datentyp: ganze Zahl

Argumente:

Argumente

Argumente	Beschreibung
field_name	Das Feld mit dem Datenbereich, der angegeben werden soll.
include_ excluded	Bei Auswahl von **True()** werden auch ausgewählte Werte mitgezählt, die durch eine Auswahl in anderen Feldern ausgeschlossen sind. Bei False oder fehlenden Werten werden die Werte nicht mitgezählt.
state_name	Der Name eines alternativen Status, der für diese Visualisierung ausgewählt wurde. Beim Argument **state_name** werden nur die Auswahlen für den angegebenen Statusnamen berücksichtigt.

Beispiele und Ergebnisse:

Im folgenden Beispiel werden drei Felder verwendet, die in verschiedene Filterfenster geladen wurden: eins für **First name**, eins für **Initials** und eins für **Has cellphone**.

Beispiele und Ergebnisse

Beispiele	Ergebnisse
Vorgabe: **John** ist in **First name** ausgewählt. `GetSelectedCount ([First name])`	1, da ein Wert in **First name** ausgewählt ist.
Vorgabe: **John** ist in **First name** ausgewählt. `GetSelectedCount ([Initials])`	0, da keine Werte in **Initials** ausgewählt sind.
Wurde keine Auswahl in **First name** getroffen, wählen Sie alle Werte in **Initials** und danach unter **Yes** den Wert **Has cellphone** aus. `GetSelectedCount ([Initials], True ())`	6. Obwohl bei den ausgewählten Optionen mit **Initials** MC und PD **Has cellphone** auf **No** gesetzt wurde, ist das Ergebnis weiterhin 6, da das Argument include_excluded auf True() gesetzt ist.

Im Beispiel verwendete Daten:

```
Names:
LOAD * inline [
First name|Last name|Initials|Has cellphone
John|Anderson|JA|Yes
Sue|Brown|SB|Yes
Mark|Carr|MC|No
Peter|Devonshire|PD|No
Jane|Elliot|JE|Yes
Peter|Franc|PF|Yes ] (delimiter is '|');
```

8.10 Dateifunktionen

Die Dateifunktionen liefern Informationen über die Datei, aus der gerade Daten geladen werden. Sie sind daher nur im Skript verfügbar. Diese Funktionen geben für alle Datenquellen außer Tabellendateien NULL zurück (Ausnahme: **ConnectString( )**).

Dateifunktionen – Übersicht

Jede Funktion wird nach der Übersicht genauer beschrieben. Sie können auch auf den Funktionsnamen in der Syntax klicken, um direkt auf die Details zu der spezifischen Funktion zuzugreifen.

Attribute

Diese Skriptfunktion liefert den Wert der Metatags unterschiedlicher Mediendateiformate als Text. Folgende Dateiformate werden unterstützt: MP3, WMA, WMV, PNG und JPG. Sollte die Datei **filename** nicht existieren, keines der unterstützten Dateiformate haben oder kein Metatag namens **attributename** enthalten, ist das Ergebnis NULL.

```
Attribute (filename, attributename)
```

ConnectString

Die Funktion **ConnectString()** gibt den Namen der aktiven Datenverbindung für ODBC- oder OLE DB-Verbindungen zurück. Die Funktion liefert einen Leer-String, wenn noch kein entsprechender **connect**-Befehl ausgeführt wurde oder wenn sie durch den Befehl **disconnect** beendet wurde.

```
ConnectString ()
```

FileBaseName

Die Funktion **FileBaseName** liefert den Namen der gerade eingelesenen Tabellendatei, ohne Pfad und Erweiterung.

```
FileBaseName ()
```

FileDir

Die Funktion **FileDir** liefert den Pfad zum Verzeichnis der gerade eingelesenen Tabellendatei.

```
FileDir ()
```

FileExtension

Die Funktion **FileExtension** liefert einen String mit der Erweiterung der gerade eingelesenen Tabellendatei.

```
FileExtension ()
```

FileName
Die Funktion **FileName** liefert den Namen der gerade eingelesenen Tabellendatei, ohne Pfad, aber mit Erweiterung.

```
FileName ()
```

FilePath
Die Funktion **FilePath** liefert den vollständigen Pfad zur gerade eingelesenen Tabellendatei.

```
FilePath ()
```

FileSize
Die Funktion **FileSize** liefert eine ganze Zahl, die die Größe der Datei filename in Byte angibt. Ist filename nicht angegeben, wird die Größe der gerade eingelesenen Tabellendatei ausgegeben.

```
FileSize ()
```

FileTime
Die Funktion **FileTime** gibt einen Zeitstempel im UTC-Format für die letzte Änderung einer angegebenen Datei zurück. Wenn keine Datei angegeben wird, gibt die Funktion einen Zeitstempel im UTC-Format für die letzte Änderung an der aktuell gelesenen Tabellendatei zurück.

```
FileTime ([ filename ])
```

GetFolderPath
Die Funktion **GetFolderPath** liefert den Wert der Microsoft Windows *SHGetFolderPath*-Funktion. Diese Funktion nimmt als Eingabe den Namen eines Microsoft Windows -Ordners und liefert den vollständigen Pfad des Ordners.

```
GetFolderPath ()
```

QvdCreateTime
Diese Skriptfunktion gibt den im XML-Header der QVD-Datei gespeicherten Zeitstempel zurück, sofern dieser in der Datei vorhanden ist, ansonsten gibt sie sie das Ergebnis NULL zurück. Im Zeitstempel wird die Uhrzeit im UTC-Format angegeben.

```
QvdCreateTime (filename)
```

QvdFieldName
Diese Skriptfunktion liefert den Namen von Feld Nummer **fieldno** in einer QVD-Datei. Ist das Feld nicht vorhanden, liefert diese Funktion NULL.

```
QvdFieldName (filename , fieldno)
```

QvdNoOfFields
Diese Skriptfunktion liefert die Zahl der Felder in einer QVD-Datei.

```
QvdNoOfFields (filename)
```

QvdNoOfRecords
Diese Skriptfunktion liefert die Zahl der Datensätze in einer QVD-Datei.

```
QvdNoOfRecords (filename)
```

QvdTableName
Diese Skriptfunktion liefert den Namen der in der QVD-Datei gespeicherten Tabelle.

```
QvdTableName (filename)
```

Attribute

Diese Skriptfunktion liefert den Wert der Metatags unterschiedlicher Mediendateiformate als Text. Folgende Dateiformate werden unterstützt: MP3, WMA, WMV, PNG und JPG. Sollte die Datei **filename** nicht existieren, keines der unterstützten Dateiformate haben oder kein Metatag namens **attributename** enthalten, ist das Ergebnis NULL.

Syntax:

```
Attribute(filename, attributename)
```

Eine große Anzahl an Metatags kann eingelesen werden. Die Beispiele in diesem Thema zeigen, welche Tags für die jeweiligen unterstützen Dateitypen eingelesen werden können.

> *Es können nur Metatags eingelesen werden, die in der Datei entsprechend dem zutreffenden Format gespeichert wurden, beispielsweise ID2v3 für MP3-Dateien oder EXIF für JPG-Dateien, jedoch keine Metadaten, die im **Datei-Explorer von Windows** gespeichert sind.*

Argumente:

Argumente

Argument	Beschreibung
filename	Der Name der Mediendatei, falls nötig mit Pfad, als Ordner-Datenverbindung. **Beispiel: *'lib://Table Files/'*** Im Legacymodus für die Skripterstellung werden die folgenden Pfadformate ebenfalls unterstützt: • absolut **Beispiel: *c:\data*** • relativ zum Qlik Sense App-Arbeitsverzeichnis. **Beispiel: *data***
attributename	Der Name eines Metatags.

Die Beispiele nutzen die Funktion **GetFolderPath**, um Pfade in den Mediendateien ausfindig zu machen. Da **GetFolderPath** nur im Legacymodus unterstützt wird, müssen Sie die Verweise auf **GetFolderPath** durch einen lib://-Datenverbindungspfad ersetzen, wenn Sie diese Funktion im Standardmodus oder in Qlik Sense SaaS verwenden.

Zugriffsbeschränkung für Dateisystem (page 1540)

Example 1: MP3-Dateien

Dieses Skript liest alle möglichen MP3-Metatags im Ordner *MyMusic*.

```
// Script to read MP3 meta tags
for each vExt in 'mp3'
for each vFoundFile in filelist( GetFolderPath('MyMusic') & '\*.'& vExt )
FileList:
LOAD FileLongName,
    subfield(FileLongName,'\',-1) as FileShortName,
    num(FileSize(FileLongName),'# ### ### ###',',',' ') as FileSize,
    FileTime(FileLongName) as FileTime,
    // ID3v1.0 and ID3v1.1 tags
    Attribute(FileLongName, 'Title') as Title,
    Attribute(FileLongName, 'Artist') as Artist,
    Attribute(FileLongName, 'Album') as Album,
    Attribute(FileLongName, 'Year') as Year,
    Attribute(FileLongName, 'Comment') as Comment,
    Attribute(FileLongName, 'Track') as Track,
    Attribute(FileLongName, 'Genre') as Genre,

    // ID3v2.3 tags
    Attribute(FileLongName, 'AENC') as AENC, // Audio encryption
    Attribute(FileLongName, 'APIC') as APIC, // Attached picture
    Attribute(FileLongName, 'COMM') as COMM, // Comments
    Attribute(FileLongName, 'COMR') as COMR, // Commercial frame
    Attribute(FileLongName, 'ENCR') as ENCR, // Encryption method registration
    Attribute(FileLongName, 'EQUA') as EQUA, // Equalization
    Attribute(FileLongName, 'ETCO') as ETCO, // Event timing codes
    Attribute(FileLongName, 'GEOB') as GEOB, // General encapsulated object
    Attribute(FileLongName, 'GRID') as GRID, // Group identification registration
    Attribute(FileLongName, 'IPLS') as IPLS, // Involved people list
    Attribute(FileLongName, 'LINK') as LINK, // Linked information
    Attribute(FileLongName, 'MCDI') as MCDI, // Music CD identifier
    Attribute(FileLongName, 'MLLT') as MLLT, // MPEG location lookup table
    Attribute(FileLongName, 'OWNE') as OWNE, // Ownership frame
    Attribute(FileLongName, 'PRIV') as PRIV, // Private frame
    Attribute(FileLongName, 'PCNT') as PCNT, // Play counter
    Attribute(FileLongName, 'POPM') as POPM, // Popularimeter

    Attribute(FileLongName, 'POSS') as POSS, // Position synchronisation frame
    Attribute(FileLongName, 'RBUF') as RBUF, // Recommended buffer size
    Attribute(FileLongName, 'RVAD') as RVAD, // Relative volume adjustment
    Attribute(FileLongName, 'RVRB') as RVRB, // Reverb
    Attribute(FileLongName, 'SYLT') as SYLT, // Synchronized lyric/text
    Attribute(FileLongName, 'SYTC') as SYTC, // Synchronized tempo codes
    Attribute(FileLongName, 'TALB') as TALB, // Album/Movie/Show title
    Attribute(FileLongName, 'TBPM') as TBPM, // BPM (beats per minute)
```

```
    Attribute(FileLongName, 'TCOM') as TCOM, // Composer
    Attribute(FileLongName, 'TCON') as TCON, // Content type
    Attribute(FileLongName, 'TCOP') as TCOP, // Copyright message
    Attribute(FileLongName, 'TDAT') as TDAT, // Date
    Attribute(FileLongName, 'TDLY') as TDLY, // Playlist delay

    Attribute(FileLongName, 'TENC') as TENC, // Encoded by
    Attribute(FileLongName, 'TEXT') as TEXT, // Lyricist/Text writer
    Attribute(FileLongName, 'TFLT') as TFLT, // File type
    Attribute(FileLongName, 'TIME') as TIME, // Time
    Attribute(FileLongName, 'TIT1') as TIT1, // Content group description
    Attribute(FileLongName, 'TIT2') as TIT2, // Title/songname/content description
    Attribute(FileLongName, 'TIT3') as TIT3, // Subtitle/Description refinement
    Attribute(FileLongName, 'TKEY') as TKEY, // Initial key
    Attribute(FileLongName, 'TLAN') as TLAN, // Language(s)
    Attribute(FileLongName, 'TLEN') as TLEN, // Length
    Attribute(FileLongName, 'TMED') as TMED, // Media type

    Attribute(FileLongName, 'TOAL') as TOAL, // Original album/movie/show title
    Attribute(FileLongName, 'TOFN') as TOFN, // Original filename
    Attribute(FileLongName, 'TOLY') as TOLY, // Original lyricist(s)/text writer(s)
    Attribute(FileLongName, 'TOPE') as TOPE, // Original artist(s)/performer(s)
    Attribute(FileLongName, 'TORY') as TORY, // Original release year
    Attribute(FileLongName, 'TOWN') as TOWN, // File owner/licensee
    Attribute(FileLongName, 'TPE1') as TPE1, // Lead performer(s)/Soloist(s)
    Attribute(FileLongName, 'TPE2') as TPE2, // Band/orchestra/accompaniment

    Attribute(FileLongName, 'TPE3') as TPE3, // Conductor/performer refinement
    Attribute(FileLongName, 'TPE4') as TPE4, // Interpreted, remixed, or otherwise modified by
    Attribute(FileLongName, 'TPOS') as TPOS, // Part of a set
    Attribute(FileLongName, 'TPUB') as TPUB, // Publisher
    Attribute(FileLongName, 'TRCK') as TRCK, // Track number/Position in set
    Attribute(FileLongName, 'TRDA') as TRDA, // Recording dates
    Attribute(FileLongName, 'TRSN') as TRSN, // Internet radio station name
    Attribute(FileLongName, 'TRSO') as TRSO, // Internet radio station owner

    Attribute(FileLongName, 'TSIZ') as TSIZ, // Size
    Attribute(FileLongName, 'TSRC') as TSRC, // ISRC (international standard recording code)
    Attribute(FileLongName, 'TSSE') as TSSE, // Software/Hardware and settings used for
encoding
    Attribute(FileLongName, 'TYER') as TYER, // Year
    Attribute(FileLongName, 'TXXX') as TXXX, // User defined text information frame
    Attribute(FileLongName, 'UFID') as UFID, // Unique file identifier
    Attribute(FileLongName, 'USER') as USER, // Terms of use
    Attribute(FileLongName, 'USLT') as USLT, // Unsychronized lyric/text transcription
    Attribute(FileLongName, 'WCOM') as WCOM, // Commercial information
    Attribute(FileLongName, 'WCOP') as WCOP, // Copyright/Legal information

    Attribute(FileLongName, 'WOAF') as WOAF, // Official audio file webpage
    Attribute(FileLongName, 'WOAR') as WOAR, // Official artist/performer webpage
    Attribute(FileLongName, 'WOAS') as WOAS, // Official audio source webpage
    Attribute(FileLongName, 'WORS') as WORS, // Official internet radio station homepage
    Attribute(FileLongName, 'WPAY') as WPAY, // Payment
    Attribute(FileLongName, 'WPUB') as WPUB, // Publishers official webpage
    Attribute(FileLongName, 'WXXX') as WXXX; // User defined URL link frame
```

```
LOAD @1:n as FileLongName Inline "$(vFoundFile)" (fix, no labels);
Next vFoundFile
Next vExt
```

Example 2: JPEG

Dieses Skript liest alle möglichen EXIF-Metatags aus JPG-Dateien im Ordner *MyPictures*.

```
// Script to read Jpeg Exif meta tags
for each vExt in 'jpg', 'jpeg', 'jpe', 'jfif', 'jif', 'jfi'
for each vFoundFile in filelist( GetFolderPath('MyPictures') & '\*.'& vExt )

FileList:
LOAD FileLongName,
    subfield(FileLongName,'\',-1) as FileShortName,
    num(FileSize(FileLongName),'# ### ### ###',',',' ') as FileSize,
    FileTime(FileLongName) as FileTime,
    // ************   Exif Main (IFD0) Attributes   ************
    Attribute(FileLongName, 'ImageWidth') as ImageWidth,
    Attribute(FileLongName, 'ImageLength') as ImageLength,
    Attribute(FileLongName, 'BitsPerSample') as BitsPerSample,
    Attribute(FileLongName, 'Compression') as Compression,

    //  examples: 1=uncompressed, 2=CCITT, 3=CCITT 3, 4=CCITT 4,

    //5=LZW, 6=JPEG (old style), 7=JPEG, 8=Deflate, 32773=PackBits RLE,
    Attribute(FileLongName, 'PhotometricInterpretation') as PhotometricInterpretation,

    //  examples: 0=WhiteIsZero, 1=BlackIsZero, 2=RGB, 3=Palette, 5=CMYK, 6=YCbCr,
    Attribute(FileLongName, 'ImageDescription') as ImageDescription,
    Attribute(FileLongName, 'Make') as Make,
    Attribute(FileLongName, 'Model') as Model,
    Attribute(FileLongName, 'StripOffsets') as StripOffsets,
    Attribute(FileLongName, 'Orientation') as Orientation,

    //  examples: 1=TopLeft, 2=TopRight, 3=BottomRight, 4=BottomLeft,

    // 5=LeftTop, 6=RightTop, 7=RightBottom, 8=LeftBottom,
    Attribute(FileLongName, 'SamplesPerPixel') as SamplesPerPixel,
    Attribute(FileLongName, 'RowsPerStrip') as RowsPerStrip,
    Attribute(FileLongName, 'StripByteCounts') as StripByteCounts,
    Attribute(FileLongName, 'XResolution') as XResolution,
    Attribute(FileLongName, 'YResolution') as YResolution,
    Attribute(FileLongName, 'PlanarConfiguration') as PlanarConfiguration,

    //  examples: 1=chunky format, 2=planar format,
    Attribute(FileLongName, 'ResolutionUnit') as ResolutionUnit,

    //  examples: 1=none, 2=inches, 3=centimeters,
    Attribute(FileLongName, 'TransferFunction') as TransferFunction,
    Attribute(FileLongName, 'Software') as Software,
    Attribute(FileLongName, 'DateTime') as DateTime,
    Attribute(FileLongName, 'Artist') as Artist,
    Attribute(FileLongName, 'HostComputer') as HostComputer,
    Attribute(FileLongName, 'WhitePoint') as WhitePoint,
```

```
    Attribute(FileLongName, 'PrimaryChromaticities') as PrimaryChromaticities,
    Attribute(FileLongName, 'YCbCrCoefficients') as YCbCrCoefficients,
    Attribute(FileLongName, 'YCbCrSubSampling') as YCbCrSubSampling,
    Attribute(FileLongName, 'YCbCrPositioning') as YCbCrPositioning,

    //  examples: 1=centered, 2=co-sited,
    Attribute(FileLongName, 'ReferenceBlackWhite') as ReferenceBlackWhite,
    Attribute(FileLongName, 'Rating') as Rating,
    Attribute(FileLongName, 'RatingPercent') as RatingPercent,
    Attribute(FileLongName, 'ThumbnailFormat') as ThumbnailFormat,

    //  examples: 0=Raw Rgb, 1=Jpeg,
    Attribute(FileLongName, 'Copyright') as Copyright,
    Attribute(FileLongName, 'ExposureTime') as ExposureTime,
    Attribute(FileLongName, 'FNumber') as FNumber,
    Attribute(FileLongName, 'ExposureProgram') as ExposureProgram,

    //  examples: 0=Not defined, 1=Manual, 2=Normal program, 3=Aperture priority, 4=Shutter
priority,

    // 5=Creative program, 6=Action program, 7=Portrait mode, 8=Landscape mode, 9=Bulb,
    Attribute(FileLongName, 'ISOSpeedRatings') as ISOSpeedRatings,
    Attribute(FileLongName, 'TimeZoneOffset') as TimeZoneOffset,
    Attribute(FileLongName, 'SensitivityType') as SensitivityType,

    //  examples: 0=Unknown, 1=Standard output sensitivity (SOS), 2=Recommended exposure index
(REI),

    // 3=ISO speed, 4=Standard output sensitivity (SOS) and Recommended exposure index (REI),

    //5=Standard output sensitivity (SOS) and ISO Speed, 6=Recommended exposure index (REI)
and ISO Speed,

    // 7=Standard output sensitivity (SOS) and Recommended exposure index (REI) and ISO speed,
    Attribute(FileLongName, 'ExifVersion') as ExifVersion,
    Attribute(FileLongName, 'DateTimeOriginal') as DateTimeOriginal,
    Attribute(FileLongName, 'DateTimeDigitized') as DateTimeDigitized,
    Attribute(FileLongName, 'ComponentsConfiguration') as ComponentsConfiguration,

    //  examples: 1=Y, 2=Cb, 3=Cr, 4=R, 5=G, 6=B,
    Attribute(FileLongName, 'CompressedBitsPerPixel') as CompressedBitsPerPixel,
    Attribute(FileLongName, 'ShutterSpeedValue') as ShutterSpeedValue,
    Attribute(FileLongName, 'ApertureValue') as ApertureValue,
    Attribute(FileLongName, 'BrightnessValue') as BrightnessValue, //  examples: -1=Unknown,
    Attribute(FileLongName, 'ExposureBiasValue') as ExposureBiasValue,
    Attribute(FileLongName, 'MaxApertureValue') as MaxApertureValue,
    Attribute(FileLongName, 'SubjectDistance') as SubjectDistance,

    //  examples: 0=Unknown, -1=Infinity,
    Attribute(FileLongName, 'MeteringMode') as MeteringMode,

    //  examples: 0=Unknown, 1=Average, 2=CenterWeightedAverage, 3=Spot,

    // 4=MultiSpot, 5=Pattern, 6=Partial, 255=Other,
    Attribute(FileLongName, 'LightSource') as LightSource,
```

```
    //  examples: 0=Unknown, 1=Daylight, 2=Fluorescent, 3=Tungsten, 4=Flash, 9=Fine weather,

    // 10=Cloudy weather, 11=Shade, 12=Daylight fluorescent,

    // 13=Day white fluorescent, 14=Cool white fluorescent,

    // 15=White fluorescent, 17=Standard light A, 18=Standard light B, 19=Standard light C,

    // 20=D55, 21=D65, 22=D75, 23=D50, 24=ISO studio tungsten, 255=other light source,
    Attribute(FileLongName, 'Flash') as Flash,
    Attribute(FileLongName, 'FocalLength') as FocalLength,
    Attribute(FileLongName, 'SubjectArea') as SubjectArea,
    Attribute(FileLongName, 'MakerNote') as MakerNote,
    Attribute(FileLongName, 'UserComment') as UserComment,
    Attribute(FileLongName, 'SubSecTime') as SubSecTime,

    Attribute(FileLongName, 'SubsecTimeOriginal') as SubsecTimeOriginal,
    Attribute(FileLongName, 'SubsecTimeDigitized') as SubsecTimeDigitized,
    Attribute(FileLongName, 'XPTitle') as XPTitle,
    Attribute(FileLongName, 'XPComment') as XPComment,

    Attribute(FileLongName, 'XPAuthor') as XPAuthor,
    Attribute(FileLongName, 'XPKeywords') as XPKeywords,
    Attribute(FileLongName, 'XPSubject') as XPSubject,
    Attribute(FileLongName, 'FlashpixVersion') as FlashpixVersion,
    Attribute(FileLongName, 'ColorSpace') as ColorSpace, //  examples: 1=sRGB,
65535=Uncalibrated,
    Attribute(FileLongName, 'PixelXDimension') as PixelXDimension,
    Attribute(FileLongName, 'PixelYDimension') as PixelYDimension,
    Attribute(FileLongName, 'RelatedSoundFile') as RelatedSoundFile,

    Attribute(FileLongName, 'FocalPlaneXResolution') as FocalPlaneXResolution,
    Attribute(FileLongName, 'FocalPlaneYResolution') as FocalPlaneYResolution,
    Attribute(FileLongName, 'FocalPlaneResolutionUnit') as FocalPlaneResolutionUnit,

    //  examples: 1=None, 2=Inch, 3=Centimeter,
    Attribute(FileLongName, 'ExposureIndex') as ExposureIndex,
    Attribute(FileLongName, 'SensingMethod') as SensingMethod,

    //  examples: 1=Not defined, 2=One-chip color area sensor, 3=Two-chip color area sensor,

    // 4=Three-chip color area sensor, 5=Color sequential area sensor,

    // 7=Trilinear sensor, 8=Color sequential linear sensor,
    Attribute(FileLongName, 'FileSource') as FileSource,

    //  examples: 0=Other, 1=Scanner of transparent type,

    // 2=Scanner of reflex type, 3=Digital still camera,
    Attribute(FileLongName, 'SceneType') as SceneType,

    //  examples: 1=A directly photographed image,
    Attribute(FileLongName, 'CFAPattern') as CFAPattern,
    Attribute(FileLongName, 'CustomRendered') as CustomRendered,
```

```
    //  examples: 0=Normal process, 1=Custom process,
    Attribute(FileLongName, 'ExposureMode') as ExposureMode,

    //  examples: 0=Auto exposure, 1=Manual exposure, 2=Auto bracket,
    Attribute(FileLongName, 'WhiteBalance') as WhiteBalance,

    //  examples: 0=Auto white balance, 1=Manual white balance,
    Attribute(FileLongName, 'DigitalZoomRatio') as DigitalZoomRatio,
    Attribute(FileLongName, 'FocalLengthIn35mmFilm') as FocalLengthIn35mmFilm,
    Attribute(FileLongName, 'SceneCaptureType') as SceneCaptureType,

    //  examples: 0=Standard, 1=Landscape, 2=Portrait, 3=Night scene,
    Attribute(FileLongName, 'GainControl') as GainControl,

    //  examples: 0=None, 1=Low gain up, 2=High gain up, 3=Low gain down, 4=High gain down,
    Attribute(FileLongName, 'Contrast') as Contrast,

    //  examples: 0=Normal, 1=Soft, 2=Hard,
    Attribute(FileLongName, 'Saturation') as Saturation,

    //  examples: 0=Normal, 1=Low saturation, 2=High saturation,
    Attribute(FileLongName, 'Sharpness') as Sharpness,

    //  examples: 0=Normal, 1=Soft, 2=Hard,
    Attribute(FileLongName, 'SubjectDistanceRange') as SubjectDistanceRange,

    //  examples: 0=Unknown, 1=Macro, 2=Close view, 3=Distant view,
    Attribute(FileLongName, 'ImageUniqueID') as ImageUniqueID,
    Attribute(FileLongName, 'BodySerialNumber') as BodySerialNumber,
    Attribute(FileLongName, 'CMNT_GAMMA') as CMNT_GAMMA,
    Attribute(FileLongName, 'PrintImageMatching') as PrintImageMatching,
    Attribute(FileLongName, 'OffsetSchema') as OffsetSchema,

    // ************   Interoperability Attributes   ************
    Attribute(FileLongName, 'InteroperabilityIndex') as InteroperabilityIndex,
    Attribute(FileLongName, 'InteroperabilityVersion') as InteroperabilityVersion,
    Attribute(FileLongName, 'InteroperabilityRelatedImageFileFormat') as
InteroperabilityRelatedImageFileFormat,
    Attribute(FileLongName, 'InteroperabilityRelatedImageWidth') as
InteroperabilityRelatedImageWidth,
    Attribute(FileLongName, 'InteroperabilityRelatedImageLength') as
InteroperabilityRelatedImageLength,
    Attribute(FileLongName, 'InteroperabilityColorSpace') as InteroperabilityColorSpace,

    //  examples: 1=sRGB, 65535=Uncalibrated,
    Attribute(FileLongName, 'InteroperabilityPrintImageMatching') as
InteroperabilityPrintImageMatching,
    // ************   GPS Attributes   ************
    Attribute(FileLongName, 'GPSVersionID') as GPSVersionID,
    Attribute(FileLongName, 'GPSLatitudeRef') as GPSLatitudeRef,
    Attribute(FileLongName, 'GPSLatitude') as GPSLatitude,
    Attribute(FileLongName, 'GPSLongitudeRef') as GPSLongitudeRef,
    Attribute(FileLongName, 'GPSLongitude') as GPSLongitude,
    Attribute(FileLongName, 'GPSAltitudeRef') as GPSAltitudeRef,
```

```
    //  examples: 0=Above sea level, 1=Below sea level,
    Attribute(FileLongName, 'GPSAltitude') as GPSAltitude,
    Attribute(FileLongName, 'GPSTimeStamp') as GPSTimeStamp,
    Attribute(FileLongName, 'GPSSatellites') as GPSSatellites,
    Attribute(FileLongName, 'GPSStatus') as GPSStatus,
    Attribute(FileLongName, 'GPSMeasureMode') as GPSMeasureMode,
    Attribute(FileLongName, 'GPSDOP') as GPSDOP,
    Attribute(FileLongName, 'GPSSpeedRef') as GPSSpeedRef,

    Attribute(FileLongName, 'GPSSpeed') as GPSSpeed,
    Attribute(FileLongName, 'GPSTrackRef') as GPSTrackRef,
    Attribute(FileLongName, 'GPSTrack') as GPSTrack,
    Attribute(FileLongName, 'GPSImgDirectionRef') as GPSImgDirectionRef,
    Attribute(FileLongName, 'GPSImgDirection') as GPSImgDirection,
    Attribute(FileLongName, 'GPSMapDatum') as GPSMapDatum,
    Attribute(FileLongName, 'GPSDestLatitudeRef') as GPSDestLatitudeRef,

    Attribute(FileLongName, 'GPSDestLatitude') as GPSDestLatitude,
    Attribute(FileLongName, 'GPSDestLongitudeRef') as GPSDestLongitudeRef,
    Attribute(FileLongName, 'GPSDestLongitude') as GPSDestLongitude,
    Attribute(FileLongName, 'GPSDestBearingRef') as GPSDestBearingRef,
    Attribute(FileLongName, 'GPSDestBearing') as GPSDestBearing,
    Attribute(FileLongName, 'GPSDestDistanceRef') as GPSDestDistanceRef,

    Attribute(FileLongName, 'GPSDestDistance') as GPSDestDistance,
    Attribute(FileLongName, 'GPSProcessingMethod') as GPSProcessingMethod,
    Attribute(FileLongName, 'GPSAreaInformation') as GPSAreaInformation,
    Attribute(FileLongName, 'GPSDateStamp') as GPSDateStamp,
    Attribute(FileLongName, 'GPSDifferential') as GPSDifferential;

    //  examples: 0=No correction, 1=Differential correction,
LOAD @1:n as FileLongName Inline "$(vFoundFile)" (fix, no labels);
Next vFoundFile
Next vExt
```

Example 3: Windows-Mediendateien

Dieses Skript liest alle möglichen WMA/WMV ASF-Metatags im Ordner *MyMusic*.

```
/ Script to read WMA/WMV ASF meta tags
for each vExt in 'asf', 'wma', 'wmv'
for each vFoundFile in filelist( GetFolderPath('MyMusic') & '\*.'& vExt )

FileList:
LOAD FileLongName,
    subfield(FileLongName,'\',-1) as FileShortName,
    num(FileSize(FileLongName),'# ### ### ###',',',' ') as FileSize,
    FileTime(FileLongName) as FileTime,
    Attribute(FileLongName, 'Title') as Title,
    Attribute(FileLongName, 'Author') as Author,
    Attribute(FileLongName, 'Copyright') as Copyright,
    Attribute(FileLongName, 'Description') as Description,

    Attribute(FileLongName, 'Rating') as Rating,
    Attribute(FileLongName, 'PlayDuration') as PlayDuration,
```

```
    Attribute(FileLongName, 'MaximumBitrate') as MaximumBitrate,
    Attribute(FileLongName, 'WMFSDKVersion') as WMFSDKVersion,
    Attribute(FileLongName, 'WMFSDKNeeded') as WMFSDKNeeded,
    Attribute(FileLongName, 'IsVBR') as IsVBR,
    Attribute(FileLongName, 'ASFLeakyBucketPairs') as ASFLeakyBucketPairs,

    Attribute(FileLongName, 'PeakValue') as PeakValue,
    Attribute(FileLongName, 'AverageLevel') as AverageLevel;
LOAD @1:n as FileLongName Inline "$(vFoundFile)" (fix, no labels);
Next vFoundFile
Next vExt
```

Example 4: PNG

Dieses Skript liest alle möglichen PNG-Metatags im Ordner *MyPictures*.

```
// Script to read PNG meta tags
for each vExt in 'png'
for each vFoundFile in filelist( GetFolderPath('MyPictures') & '\*.'& vExt )

FileList:
LOAD FileLongName,
    subfield(FileLongName,'\',-1) as FileShortName,
    num(FileSize(FileLongName),'# ### ### ###',',',' ') as FileSize,
    FileTime(FileLongName) as FileTime,
    Attribute(FileLongName, 'Comment') as Comment,

    Attribute(FileLongName, 'Creation Time') as Creation_Time,
    Attribute(FileLongName, 'Source') as Source,
    Attribute(FileLongName, 'Title') as Title,
    Attribute(FileLongName, 'Software') as Software,
    Attribute(FileLongName, 'Author') as Author,
    Attribute(FileLongName, 'Description') as Description,

    Attribute(FileLongName, 'Copyright') as Copyright;
LOAD @1:n as FileLongName Inline "$(vFoundFile)" (fix, no labels);
Next vFoundFile
Next vExt
```

ConnectString

Die Funktion **ConnectString()** gibt den Namen der aktiven Datenverbindung für ODBC- oder OLE DB-Verbindungen zurück. Die Funktion liefert einen Leer-String, wenn noch kein entsprechender **connect**-Befehl ausgeführt wurde oder wenn sie durch den Befehl **disconnect** beendet wurde.

Syntax:

```
ConnectString()
```

Beispiele und Ergebnisse:

Skriptbeispiele

Beispiel	Ergebnis

`LIB CONNECT TO 'Tutorial ODBC';`
 	
`ConnectString:`
 	
`Load ConnectString() as ConnectString AutoGenerate 1;`
	Liefert „Tutorial ODBC“ im Feld ConnectString.
	
	Dieses Beispiel nimmt an, dass eine verfügbare Datenverbindung namens Tutorial ODBC besteht.

FileBaseName

Die Funktion **FileBaseName** liefert den Namen der gerade eingelesenen Tabellendatei, ohne Pfad und Erweiterung.

Syntax:

```
FileBaseName()
```

Beispiele und Ergebnisse:

Skriptbeispiele

Beispiel	Ergebnis

`LOAD *, filebasename( ) as X from`
 	
`C:\UserFiles\abc.txt`
	Liefert in allen Datensätzen im Feld X den Wert „abc“.

FileDir

Die Funktion **FileDir** liefert den Pfad zum Verzeichnis der gerade eingelesenen Tabellendatei.

Syntax:

```
FileDir()
```

Diese Funktion unterstützt im Standardmodus nur Ordner-Datenverbindungen.

Beispiele und Ergebnisse:

Skriptbeispiele

Beispiel	Ergebnis

`Load *, filedir( ) as X from`
 	
`C:\UserFiles\abc.txt`
	Liefert in allen Datensätzen im Feld C:\UserFiles den Wert 'X'.

FileExtension

Die Funktion **FileExtension** liefert einen String mit der Erweiterung der gerade eingelesenen Tabellendatei.

Syntax:

```
FileExtension()
```

Beispiele und Ergebnisse:

Skriptbeispiele

Beispiel	Ergebnis

`LOAD *, FileExtension( ) as X from`
 	
`C:\UserFiles\abc.txt`
	Liefert in allen Datensätzen im Feld X den Wert „txt“.

FileName

Die Funktion **FileName** liefert den Namen der gerade eingelesenen Tabellendatei, ohne Pfad, aber mit Erweiterung.

Syntax:

```
FileName()
```

Beispiele und Ergebnisse:

Skriptbeispiele

Beispiel	Ergebnis

`LOAD *, FileName( ) as X from`
 	
`C:\UserFiles\abc.txt`
	Liefert in allen Datensätzen im Feld 'abc.txt' den Wert 'X'.

FilePath

Die Funktion **FilePath** liefert den vollständigen Pfad zur gerade eingelesenen Tabellendatei.

Syntax:

```
FilePath()
```

Diese Funktion unterstützt im Standardmodus nur Ordner-Datenverbindungen.

Beispiele und Ergebnisse:

Skriptbeispiele

Beispiel	Ergebnis
`Load *, FilePath( ) as X from C:\UserFiles\abc.txt`	Liefert in allen Datensätzen im Feld 'C:\UserFiles\abc.txt' den Wert 'X'.

FileSize

Die Funktion **FileSize** liefert eine ganze Zahl, die die Größe der Datei filename in Byte angibt. Ist filename nicht angegeben, wird die Größe der gerade eingelesenen Tabellendatei ausgegeben.

Syntax:
FileSize([filename]**)**

Argumente:

Argumente

Argument	Beschreibung
filename	Der Name einer Datei, erforderlichenfalls einschließlich des Dateipfads, als Ordner- oder Webdatei-Datenverbindung. Wenn Sie keinen Dateinamen angeben, wird die aktuell gelesene Tabellendatei verwendet. **Beispiel: *'lib://Table Files/'*** Im Legacymodus für die Skripterstellung werden die folgenden Pfadformate ebenfalls unterstützt: • absolut **Beispiel: *c:\data*** • relativ zum Qlik Sense App-Arbeitsverzeichnis. **Beispiel: *data*** • als URL-Adresse (HTTP oder FTP), die eine Datei im Internet oder Intranet lokalisiert. **Beispiel: *http://www.qlik.com***

Beispiele und Ergebnisse:

Skriptbeispiele

Beispiel	Ergebnis
`LOAD *, FileSize( ) as X from abc.txt;`	Liefert in allen Datensätzen im Feld X die Größe der angegebenen Datei (abc.txt) als ganze Zahl.
`FileSize( 'lib://DataFiles/xyz.xls' )`	Liefert die Größe der Datei xyz.xls.

FileTime

Die Funktion **FileTime** gibt einen Zeitstempel im UTC-Format für die letzte Änderung einer angegebenen Datei zurück. Wenn keine Datei angegeben wird, gibt die Funktion einen Zeitstempel im UTC-Format für die letzte Änderung an der aktuell gelesenen Tabellendatei zurück.

Syntax:

```
FileTime([ filename ])
```

Argumente:

Argumente

Argument	Beschreibung
filename	Der Name einer Datei, falls nötig einschließlich Pfad, als Ordner- oder Web-Datei-Datenverbindung. **Beispiel: *'lib://Table Files/'*** Im Legacymodus für die Skripterstellung werden die folgenden Pfadformate ebenfalls unterstützt: • absolut **Beispiel: *c:\data*** • relativ zum Qlik Sense App-Arbeitsverzeichnis. **Beispiel: *data*** • als URL-Adresse (HTTP oder FTP), die eine Datei im Internet oder Intranet lokalisiert. **Beispiel: *http://www.qlik.com***

Beispiele und Ergebnisse:

Skriptbeispiele

Beispiel	Ergebnis
`LOAD *, FileTime( ) as X from abc.txt;`	Gibt in allen Datensätzen im Feld X den Zeitstempel der letzten Änderung der Datei (abc.txt) zurück.
`FileTime( 'xyz.xls' )`	Liefert Datum und Uhrzeit der letzten Änderung der Datei xyz.xls.

GetFolderPath

Die Funktion **GetFolderPath** liefert den Wert der Microsoft Windows *SHGetFolderPath*-Funktion. Diese Funktion nimmt als Eingabe den Namen eines Microsoft Windows -Ordners und liefert den vollständigen Pfad des Ordners.

Diese Funktion wird im Standardmodus nicht unterstützt.

Syntax:

```
GetFolderPath(foldername)
```

Argumente:

Argumente

Argument	Beschreibung
foldername	Name des Microsoft Windows-Ordners. Der Ordnername darf keine Leerzeichen enthalten. Leerzeichen im Ordnernamen in Windows Explorer müssen entfernt werden. Beispiele: *MyMusic* *MyDocuments*

Beispiele und Ergebnisse:

Ziel dieses Beispiels ist es, die Pfade der folgenden Microsoft Windows-Ordner zu erhalten: *MyMusic*, *MyPictures* und *Windows*. Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus.

```
LOAD
 GetFolderPath('MyMusic') as MyMusic,
 GetFolderPath('MyPictures') as MyPictures,
 GetFolderPath('Windows') as Windows
AutoGenerate 1;
```

Sobald die App erneut ausgeführt wird, werden die Felder *MyMusic*, *MyPictures* und *Windows* zum Datenmodell hinzugefügt. Jedes Feld enthält den Pfad zum bei der Eingabe definierten Ordner. Hier ein Beispiel:

- *C:\Users\smu\Music* for the folder *MyMusic*
- *C:\Users\smu\Pictures* for the folder *MyPictures*
- *C:\Windows* for the folder *Windows*

QvdCreateTime

Diese Skriptfunktion gibt den im XML-Header der QVD-Datei gespeicherten Zeitstempel zurück, sofern dieser in der Datei vorhanden ist, ansonsten gibt sie sie das Ergebnis NULL zurück. Im Zeitstempel wird die Uhrzeit im UTC-Format angegeben.

Syntax:

```
QvdCreateTime(filename)
```

Argumente:

Argumente

Argument	Beschreibung
filename	Der Name einer QVD-Datei, falls nötig einschließlich Pfad, als Ordner- oder Web-Datenverbindung. **Beispiel: *'lib://Table Files/'*** Im Legacymodus für die Skripterstellung werden die folgenden Pfadformate ebenfalls unterstützt: • absolut **Beispiel: *c:\data*** • relativ zum Qlik Sense App-Arbeitsverzeichnis. **Beispiel: *data*** • als URL-Adresse (HTTP oder FTP), die eine Datei im Internet oder Intranet lokalisiert. **Beispiel: *http://www.qlik.com***

Beispiel:

```
QvdCreateTime('MyFile.qvd')

QvdCreateTime('C:\MyDir\MyFile.qvd')

QvdCreateTime('lib://DataFiles/MyFile.qvd')
```

QvdFieldName

Diese Skriptfunktion liefert den Namen von Feld Nummer **fieldno** in einer QVD-Datei. Ist das Feld nicht vorhanden, liefert diese Funktion NULL.

Syntax:

```
QvdFieldName(filename , fieldno)
```

Argumente:

Argumente

Argument	Beschreibung
filename	Der Name einer QVD-Datei, falls nötig einschließlich Pfad, als Ordner- oder Web-Datenverbindung. **Beispiel: *'lib://Table Files/'*** Im Legacymodus für die Skripterstellung werden die folgenden Pfadformate ebenfalls unterstützt: • absolut **Beispiel: *c:\data*** • relativ zum Qlik Sense App-Arbeitsverzeichnis. **Beispiel: *data*** • als URL-Adresse (HTTP oder FTP), die eine Datei im Internet oder Intranet lokalisiert. **Beispiel: *http://www.qlik.com***
fieldno	Die Nummer des Felds innerhalb der Tabelle in der QVD-Datei.

Beispiele:

```
QvdFieldName ('MyFile.qvd', 5)

QvdFieldName ('C:\MyDir\MyFile.qvd', 5)

QvdFieldName ('lib://DataFiles/MyFile.qvd', 5)
```

Alle drei Beispiele liefern den Namen des fünften Felds der Tabelle, die in der QVD-Datei enthalten ist.

QvdNoOfFields

Diese Skriptfunktion liefert die Zahl der Felder in einer QVD-Datei.

Syntax:

```
QvdNoOfFields(filename)
```

Argumente:

Argumente

Argument	Beschreibung
filename	Der Name einer QVD-Datei, falls nötig einschließlich Pfad, als Ordner- oder Web-Datenverbindung. **Beispiel: *'lib://Table Files/'*** Im Legacymodus für die Skripterstellung werden die folgenden Pfadformate ebenfalls unterstützt: • absolut **Beispiel: *c:\data*** • relativ zum Qlik Sense App-Arbeitsverzeichnis. **Beispiel: *data*** • als URL-Adresse (HTTP oder FTP), die eine Datei im Internet oder Intranet lokalisiert. **Beispiel: *http://www.qlik.com***

Beispiele:

```
QvdNoOfFields ('MyFile.qvd')

QvdNoOfFields ('C:\MyDir\MyFile.qvd')

QvdNoOfFields ('lib://DataFiles/MyFile.qvd')
```

QvdNoOfRecords

Beispiel: Diese Skriptfunktion liefert die Zahl der Datensätze in einer QVD-Datei.

Syntax:

```
QvdNoOfRecords(filename)
```

Argumente:

Argumente

Argument	Beschreibung
filename	Der Name einer QVD-Datei, falls nötig einschließlich Pfad, als Ordner- oder Web-Datenverbindung. **Beispiel: *'lib://Table Files/'*** Im Legacymodus für die Skripterstellung werden die folgenden Pfadformate ebenfalls unterstützt: • absolut **Beispiel: *c:\data*** • relativ zum Qlik Sense App-Arbeitsverzeichnis. **Beispiel: *data*** • als URL-Adresse (HTTP oder FTP), die eine Datei im Internet oder Intranet lokalisiert. **Beispiel: *http://www.qlik.com***

Beispiele:

```
QvdNoOfRecords ('MyFile.qvd')

QvdNoOfRecords ('C:\MyDir\MyFile.qvd')

QvdNoOfRecords ('lib://DataFiles/MyFile.qvd')
```

QvdTableName

Diese Skriptfunktion liefert den Namen der in der QVD-Datei gespeicherten Tabelle.

Syntax:

```
QvdTableName(filename)
```

Argumente:

Argumente

Argument	Beschreibung
filename	Der Name einer QVD-Datei, falls nötig einschließlich Pfad, als Ordner- oder Web-Datenverbindung. **Beispiel: *'lib://Table Files/'*** Im Legacymodus für die Skripterstellung werden die folgenden Pfadformate ebenfalls unterstützt: • absolut **Beispiel: *c:\data*** • relativ zum Qlik Sense App-Arbeitsverzeichnis. **Beispiel: *data*** • als URL-Adresse (HTTP oder FTP), die eine Datei im Internet oder Intranet lokalisiert. **Beispiel: *http://www.qlik.com***

Beispiele:

```
QvdTableName ('MyFile.qvd')

QvdTableName ('C:\MyDir\MyFile.qvd')

QvdTableName ('lib://data\MyFile.qvd')
```

8.11 Finanzfunktionen

Finanzfunktionen können sowohl im Datenladeskript als auch in den Diagrammformeln zur Berechnung von Zahlungen und Zinssätzen verwendet werden.
Zu zahlende Beträge sind stets durch ein negatives Vorzeichen gekennzeichnet. Eingehende Beträge dagegen sind positive Zahlen.
Die Finanzfunktionen (mit Ausnahme der Funktionen, deren Name mit **range**- beginnt) haben folgende Argumente.

*Bei den Finanzfunktionen ist es von besonderer Wichtigkeit, dass Sie beim Festlegen der Einheiten für **rate** und **nper** die Konsistenz berücksichtigen. Wenn Sie einen Kredit mit einer Laufzeit von 5 Jahren und einem jährlichen Zinssatz von 6 % in monatlichen Raten zurückzahlen, benutzen Sie für **rate** 0,005 (6 %/12) und für **nper** 60 (5*12). Wenn Sie denselben Kredit in jährlichen Raten zurückzahlen, benutzen Sie für **rate** 6 % und für **nper** 5.*

Finanzfunktionen – Übersicht

Jede Funktion wird nach der Übersicht genauer beschrieben. Sie können auch auf den Funktionsnamen in der Syntax klicken, um direkt auf die Details zu der spezifischen Funktion zuzugreifen.

FV
Diese Funktion liefert den zukünftigen Wert einer Investition mit konstanten Raten in konstanten Zeiträumen und mit einfachem, jährlichem Zinssatz.

```
FV (rate, nper, pmt [ ,pv [ , type ] ])
```

nPer
Diese Funktion liefert die Zahl der Raten bei einer Investition mit konstanten Raten in konstanten Zeiträumen und konstantem Zinssatz.

```
nPer (rate, pmt, pv [ ,fv [ , type ] ])
```

Pmt
Diese Funktion liefert den Betrag der Raten bei einem Darlehen mit konstanten Raten in konstanten Zeiträumen und mit konstantem Zinssatz. Diese Größe ist über den gesamten Rückzahlungszeitraum konstant. Eine Zahlung wird als negative Zahl, beispielsweise -20, angegeben.

```
Pmt (rate, nper, pv [ ,fv [ , type ] ] )
```

PV
Diese Funktion liefert den momentanen Wert einer Investition.

```
PV (rate, nper, pmt [ ,fv [ , type ] ])
```

Rate
Diese Funktion liefert den Zinssatz pro Zeitraum einer Rate. Das Ergebnis hat entsprechend dem Standardformat **Fix** zwei Dezimalstellen und ein %-Zeichen.

```
Rate (nper, pmt , pv [ ,fv [ , type ] ])
```

BlackAndSchole

Das Black and Scholes-Modell ist ein mathematisches Modell für derivative Finanzinstrumente. Die Formel berechnet den theoretischen Wert einer Option. Die **BlackAndSchole**-Funktion in Qlik Sense liefert den theoretischen Wert einer Option gemäß der unveränderten Black and Scholes-Formel (europäische Option).

```
BlackAndSchole(strike , time_left , underlying_price , vol , risk_free_rate , type)
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
strike	Der zukünftige Kaufpreis an der Börse.
time_left	Die Zahl der verbleibenden Zeiteinheiten.
underlying_ price	Der gegenwärtige Börsenwert.
vol	Volatilität (des Börsenkurses), ausgedrückt als Prozentsatz im Dezimalformat, pro Zeiteinheit.
risk_free_rate	Der risikofreie Zinssatz, ausgedrückt als Prozentsatz im Dezimalformat, pro Zeiteinheit.
call_or_put	Die Art der Option: 'c', 'call' oder ein beliebiger numerischer Wert ungleich 0 und für Verkaufsoptionen 'p', 'put' oder 0.

Beschränkungen:

Der Wert von strike, time_left und underlying_price muss >0 sein.

Der Wert von vol und risk_free_rate muss <0 oder >0 sein.

Beispiele und Ergebnisse:

Skriptbeispiele

Beispiel	Ergebnis
`BlackAndScholes(130, 4, 68.5, 0.4, 0.04, 'call')` Dies berechnet den theoretischen Wert einer Option, die gegenwärtig mit 68.5 notiert ist und in 4 Jahren zu einem Kurs von 130 gekauft wird. Die Formel verwendet eine Volatilität von 0,4 (40 %) pro Jahr, und es wird von einem risikofreien Zinssatz von 0,04 (4 %) ausgegangen.	Liefert 11,245

FV

Diese Funktion liefert den zukünftigen Wert einer Investition mit konstanten Raten in konstanten Zeiträumen und mit einfachem, jährlichem Zinssatz.

Syntax:

```
FV(rate, nper, pmt [ ,pv [ , type ] ])
```

Rückgabe Datentyp: numerisch. Standardmäßig wird das Ergebnis als Währung formatiert..

Argumente:

Argumente

Argument	Beschreibung
rate	Der Zinssatz pro Zeitraum.
nper	Die Gesamtzahl der Zahlungszeiträume der Annuitäten.
pmt	Der Zahlungsbetrag in jedem Zeitraum. Diese Größe ist über den gesamten Rückzahlungszeitraum konstant. Eine Zahlung wird als negative Zahl, beispielsweise -20, angegeben.
pv	Der Barwert einer Reihe künftiger Zahlungen. Fehlt **pv**, wird 0 (Null) angenommen.
type	Ist 0, wenn die Zahlung am Ende des Zeitraums fällig wird, oder 1, wenn die Zahlung zu Beginn des Zeitraums fällig wird. Fehlt **type**, wird 0 angenommen.

Beispiele und Ergebnisse:

Skriptbeispiel

Beispiel	Ergebnis
Angenommen, Sie bezahlen ein neues Haushaltsgerät in 36 Monatsraten von je $20. Der jährliche Zinssatz beträgt 6%. Die Rate wird jeweils am Ende des Monats fällig. Wie viel haben Sie nach Zahlung der letzten Rechnung insgesamt bezahlt? `FV(0.005,36,-20)`	Liefert $786.72

nPer

Diese Funktion liefert die Zahl der Raten bei einer Investition mit konstanten Raten in konstanten Zeiträumen und konstantem Zinssatz.

Syntax:

```
nPer(rate, pmt, pv [ ,fv [ , type ] ])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
rate	Der Zinssatz pro Zeitraum.
nper	Die Gesamtzahl der Zahlungszeiträume der Annuitäten.

Argument	Beschreibung
pmt	Der Zahlungsbetrag in jedem Zeitraum. Diese Größe ist über den gesamten Rückzahlungszeitraum konstant. Eine Zahlung wird als negative Zahl, beispielsweise -20, angegeben.
pv	Der Barwert einer Reihe künftiger Zahlungen. Fehlt **pv**, wird 0 (Null) angenommen.
fv	Der Endwert bzw. Kontostand, der mit der letzten Zahlung erreicht sein soll. Fehlt **fv**, wird 0 angenommen.
type	Ist 0, wenn die Zahlung am Ende des Zeitraums fällig wird, oder 1, wenn die Zahlung zu Beginn des Zeitraums fällig wird. Fehlt **type**, wird 0 angenommen.

Beispiele und Ergebnisse:

Skriptbeispiel

Beispiel	Ergebnis
Sie wollen ein Haushaltsgerät verkaufen und verlangen dafür monatliche Raten von 20 $. Der jährliche Zinssatz beträgt 6%. Die Rate wird jeweils am Ende des Monats fällig. Wie viele Raten müssen vereinbart werden, wenn am Ende ein Gesamtbetrag von $800 bezahlt sein soll? `nPer(0.005,-20,0,800)`	Liefert 36,56

Pmt

Diese Funktion liefert den Betrag der Raten bei einem Darlehen mit konstanten Raten in konstanten Zeiträumen und mit konstantem Zinssatz. Diese Größe ist über den gesamten Rückzahlungszeitraum konstant. Eine Zahlung wird als negative Zahl, beispielsweise -20, angegeben.

```
Pmt(rate, nper, pv [ ,fv [ , type ] ] )
```

Rückgabe Datentyp: numerisch. Standardmäßig wird das Ergebnis als Währung formatiert..

Um den gezahlten Gesamtbetrag zu ermitteln, multiplizieren Sie das Ergebnis von **pmt** mit **nper**.

Argumente:

Argumente

Argument	Beschreibung
rate	Der Zinssatz pro Zeitraum.
nper	Die Gesamtzahl der Zahlungszeiträume der Annuitäten.
pv	Der Barwert einer Reihe künftiger Zahlungen. Fehlt **pv**, wird 0 (Null) angenommen.
fv	Der Endwert bzw. Kontostand, der mit der letzten Zahlung erreicht sein soll. Fehlt **fv**, wird 0 angenommen.

Argument	Beschreibung
type	Ist 0, wenn die Zahlung am Ende des Zeitraums fällig wird, oder 1, wenn die Zahlung zu Beginn des Zeitraums fällig wird. Fehlt **type**, wird 0 angenommen.

Beispiele und Ergebnisse:

Skriptbeispiele

Beispiel	Ergebnis
Die folgende Funktion berechnet die Tilgungsraten eines Darlehens von 20.000 € mit einem jährlichen Zinssatz von 10 %, das in 8 Monatsraten zurückgezahlt wird: `Pmt(0.1/12,8,20000)`	Liefert -$2,594.66
Werden die Raten für dasselbe Darlehen zu Beginn des Monats fällig, ergibt sich folgendes Ergebnis: `Pmt(0.1/12,8,20000,0,1)`	Liefert -$2,573.21

PV

Diese Funktion liefert den momentanen Wert einer Investition.

```
PV(rate, nper, pmt [ ,fv [ , type ] ])
```

Rückgabe Datentyp: numerisch. Standardmäßig wird das Ergebnis als Währung formatiert..

Der Barwert ist die Summe, die zukünftige Zahlungen zum jetzigen Zeitpunkt wert sind. Wenn Sie beispielsweise Geld aufnehmen, ist der aufgenommene Betrag der Barwert für den Geldgeber.

Argumente:

Argumente

Argument	Beschreibung
rate	Der Zinssatz pro Zeitraum.
nper	Die Gesamtzahl der Zahlungszeiträume der Annuitäten.
pmt	Der Zahlungsbetrag in jedem Zeitraum. Diese Größe ist über den gesamten Rückzahlungszeitraum konstant. Eine Zahlung wird als negative Zahl, beispielsweise -20, angegeben.
fv	Der Endwert bzw. Kontostand, der mit der letzten Zahlung erreicht sein soll. Fehlt **fv**, wird 0 angenommen.
type	Ist 0, wenn die Zahlung am Ende des Zeitraums fällig wird, oder 1, wenn die Zahlung zu Beginn des Zeitraums fällig wird. Fehlt **type**, wird 0 angenommen.

Beispiele und Ergebnisse:

Skriptbeispiel

Beispiel	Ergebnis
Was ist der derzeitige Wert einer Verbindlichkeit, bei der Sie fünf Jahre lang zum Monatsende $ 100 bezahlen und ein jährlicher Zinssatz von 7 % zugrunde liegt? `PV(0.07/12,12*5,-100,0,0)`	Liefert $5,050.20

Rate

Diese Funktion liefert den Zinssatz pro Zeitraum einer Rate. Das Ergebnis hat entsprechend dem Standardformat **Fix** zwei Dezimalstellen und ein %-Zeichen.

Syntax:

```
Rate(nper, pmt , pv [ ,fv [ , type ] ])
```

Rückgabe Datentyp: numerisch.

Die Funktion **rate** wird anhand von Iterationen berechnet und kann keine oder mehrere Lösungen liefern. Sind die folgenden Ergebnisse von **rate** nicht konvergent, wird NULL ausgegeben.

Argumente:

Argumente

Argument	Beschreibung
nper	Die Gesamtzahl der Zahlungszeiträume der Annuitäten.
pmt	Der Zahlungsbetrag in jedem Zeitraum. Diese Größe ist über den gesamten Rückzahlungszeitraum konstant. Eine Zahlung wird als negative Zahl, beispielsweise -20, angegeben.
pv	Der Barwert einer Reihe künftiger Zahlungen. Fehlt **pv**, wird 0 (Null) angenommen.
fv	Der Endwert bzw. Kontostand, der mit der letzten Zahlung erreicht sein soll. Fehlt **fv**, wird 0 angenommen.
type	Ist 0, wenn die Zahlung am Ende des Zeitraums fällig wird, oder 1, wenn die Zahlung zu Beginn des Zeitraums fällig wird. Fehlt **type**, wird 0 angenommen.

Beispiele und Ergebnisse:

Skriptbeispiel

Beispiel	Ergebnis
Welcher monatliche Zinssatz liegt bei einem Kredit von 10.000 € mit einer Laufzeit von 5 Jahren und monatlichen Tilgungsraten von 300 € zugrunde? `Rate(60,-300,10000)`	Liefert 2.00%

8.12 Formatfunktionen

Die Formatfunktionen wenden das Anzeigeformat auf die numerischen Eingabefelder oder Formeln an. Abhängig vom Datentyp können Sie die Zeichen für das Dezimaltrennzeichen, Tausendertrennzeichen usw. definieren.

Die Funktionen geben alle einen zweifachen Wert mit dem String und dem Zahlenwert zurück. Die Funktionsweise entspricht aber einer Umwandlung von Zahlen in Strings. **Dual()** ist ein Sonderfall, aber die anderen Formatfunktionen nehmen den numerischen Wert der Formel und generieren einen String, der die Zahl repräsentiert.

Die Interpretationsfunktionen gehen genau andersherum vor: sie werten Stringformeln als Zahlen aus und geben das Format der resultierenden Zahl an.

Die Funktionen können sowohl im Datenladeskript als auch in den Diagrammformeln verwendet werden.

Bei der Zahlendarstellung wird immer ein Punkt als Dezimaltrennzeichen verwendet.

Formatfunktionen – Übersicht

Jede Funktion wird nach der Übersicht genauer beschrieben. Sie können auch auf den Funktionsnamen in der Syntax klicken, um direkt auf die Details zu der spezifischen Funktion zuzugreifen.

ApplyCodepage
ApplyCodepage() wendet einen anderen Codepage-Zeichensatz auf das im Ausdruck angegebene Feld oder den Text an. Das **codepage**-Argument muss numerisch sein.

```
ApplyCodepage (text, codepage)
```

Date
Date() formatiert eine Formel als Datum und verwendet dafür das vorgegebene Format aus den Systemvariablen des Datenladeskripts, des Betriebssystems oder eines Formatcodes, wenn dieser zur Verfügung steht.

```
Date (number[, format])
```

Dual
Dual() kombiniert eine Zahl und einen String in einem einzelnen Datensatz, sodass die Zahlendarstellung des Datensatzes zur Sortierung und Berechnung verwendet und der Stringwert zu Darstellungszwecken genutzt werden kann.

```
Dual (text, number)
```

Interval

Interval() formatiert eine Zahl als Zeitintervall und verwendet dafür das vorgegebene Format aus den Systemvariablen des Datenladeskripts, des Betriebssystems oder eines Formatcodes, wenn dieser zur Verfügung steht.

```
Interval (number[, format])
```

Money

Money() formatiert eine Formel numerisch als Geldwert und verwendet dafür, wenn kein Formatcode zur Verfügung steht, das vorgegebene Format aus den Systemvariablen des Datenladeskripts oder des Betriebssystems sowie optionale Dezimal- und Tausendertrennzeichen.

```
Money (number[, format[, dec_sep [, thou_sep]]])
```

Num

Num() formatiert eine Zahl, konvertiert also den Zahlenwert der Eingabe in den Eingabetext mit dem Format, das im zweiten Parameter angegeben ist. Wenn der zweite Parameter ausgelassen wird, werden die Dezimal- und Tausendertrennzeichen verwendet, die im Datenladeskript festgelegt sind. Angepasste Dezimal- und Tausendertrennzeichen sind optionale Parameter.

```
Num (number[, format[, dec_sep [, thou_sep]]])
```

Time

Time() formatiert eine Formel als Zeitwert und verwendet dafür, wenn kein Formatcode zur Verfügung steht, das vorgegebene Format aus den Systemvariablen des Datenladeskripts oder des Betriebssystems.

```
Time (number[, format])
```

Timestamp

TimeStamp() formatiert eine Formel als Datum- und Zeitwert und verwendet dafür, wenn kein Formatcode zur Verfügung steht, das vorgegebene Zeitstempelformat aus den Systemvariablen des Datenladeskripts oder des Betriebssystems.

```
Timestamp (number[, format])
```

Siehe auch:

Interpretationsfunktionen (page 1291)

ApplyCodepage

ApplyCodepage() wendet einen anderen Codepage-Zeichensatz auf das im Ausdruck angegebene Feld oder den Text an. Das **codepage**-Argument muss numerisch sein.

ApplyCodepage kann in Diagrammausdrücken verwendet werden, wird jedoch häufiger als Skript-Funktion im Dateneditor verwendet. Wenn Sie beispielsweise Dateien laden, die möglicherweise in unterschiedlichen Zeichensätzen gespeichert sind, die Sie nicht steuern können, dann wenden Sie die Codepage entsprechend des von Ihnen benötigten Zeichensatzes adn.

Syntax:

```
ApplyCodepage(text, codepage)
```

Rückgabe Datentyp: String

Argumente:

Argumente

Argument	Beschreibung
text	Das Feld oder der Text, das/den Sie auf eine andere Codepage anwenden möchten, wird durch das Argument **codepage** festgelegt.
codepage	Die Zahl für die Codepage, die auf das Feld oder den Ausdruck entsprechend der Festlegung in **text** angewendet wird.

Beispiele und Ergebnisse:

Skriptbeispiele

Beispiel	Ergebnis
`LOAD ApplyCodepage (ROWX,1253) as GreekProduct, ApplyCodepage (ROWY, 1255) as HebrewProduct, ApplyCodepage (ROWZ, 65001) as EnglishProduct; SQL SELECT ROWX, ROWY, ROWZ From Products;`	Beim Laden aus SQL können die Zeichensätze in der Quelle gemischt sein: kyrillisch, hebräisch usw. aus dem UTF-8-Format. Diese Zeichensätze müssen Zeile für Zeile geladen werden, wobei für jede Zeile eine andere Codepage angewendet wird. Der **codepage**-Wert 1253 repräsentiert den griechischen Windows-Zeichensatz, der Wert 1255 steht für Hebräisch und der Wert 65001 für lateinische UTF-8-Zeichen.

Siehe auch: *Zeichensatz (page 172)*

Date

Date() formatiert eine Formel als Datum und verwendet dafür das vorgegebene Format aus den Systemvariablen des Datenladeskripts, des Betriebssystems oder eines Formatcodes, wenn dieser zur Verfügung steht.

Syntax:

```
Date(number[, format])
```

Rückgabe Datentyp: dual

Argumente:

Argumente

Argument	Beschreibung
number	Die zu formatierende Zahl.
format	String, der das Format des entstehenden Strings beschreibt. Wenn kein Formatstring geliefert wird, wird das in den Systemvariablen des Datenladeskripts oder im Betriebssystem festgelegte Datumsformat verwendet.

Beispiele und Ergebnisse:
Bei den nachstehenden Beispielen gehen wir von folgenden Standardeinstellungen aus:

- Datumsformat 1: YY-MM-DD
- Datumsformat 2: M/D/YY

Beispiel:

```
Date( A )
```
dabei gilt A=35648

Ergebnistabelle

Results	Standardformat 1	Standardformat 2
String:	97-08-06	8/6/97
Zahl:	35648	35648

Beispiel:

```
Date( A, 'YY.MM.DD' )
```
dabei gilt A=35648

Ergebnistabelle

Results	Standardformat 1	Standardformat 2
String:	97.08.06	97.08.06
Zahl:	35648	35648

Beispiel:

```
Date( A, 'DD.MM.YYYY' )
```
dabei gilt A=35648.375

Ergebnistabelle

Results	Standardformat 1	Standardformat 2
String:	06.08.1997	06.08.1997
Zahl:	35648.375	35648.375

Beispiel:

```
Date( A, 'YY.MM.DD' )
```
dabei gilt A=8/6/97

Ergebnistabelle

Results	Standardformat 1	Standardformat 2
String:	NULL (leer)	97.08.06
Zahl:	NULL	35648

Dual

Dual() kombiniert eine Zahl und einen String in einem einzelnen Datensatz, sodass die Zahlendarstellung des Datensatzes zur Sortierung und Berechnung verwendet und der Stringwert zu Darstellungszwecken genutzt werden kann.

Syntax:

```
Dual(text, number)
```

Rückgabe Datentyp: dual

Argumente:

Argumente

Argument	Beschreibung
text	Der Stringwert, der in Verbindung mit dem Zahlenargument verwendet werden soll.
number	Die Zahl, die in Verbindung mit dem String im Stringargument verwendet werden soll.

In Qlik Sense sind alle Feldwerte potenziell duale Werte. Die Feldwerte können demnach sowohl einen numerischen als auch einen Textwert haben. Z. B. kann ein Datum einen numerischen Wert von 40908 und den Textwert von '2011-12-31' haben.

Wenn alle Werte dieselbe Zahlendarstellung, aber unterschiedliche Stringdarstellungen haben, nehmen alle die Stringdarstellung des ersten dieser Werte an.

*Die **dual**-Funktion steht meist am Anfang des Skripts, bevor andere Werte in das betreffende Feld geladen werden, denn auf diese Weise kann die Stringdarstellung für Filterfenster festgelegt werden.*

Beispiele und Ergebnisse:

Skriptbeispiele

Beispiel	**Beschreibung**

Fügen Sie Ihrem Skript die folgenden Beispiele hinzu und führen Sie es aus.
`Load dual ( NameDay,NumDay ) as DayOfWeek inline`
`[ NameDay,NumDay`
`Monday,0`
`Tuesday,1`
`Wednesday,2`
`Thursday,3`
`Friday,4`
`Saturday,5`
`Sunday,6 ];`
	Das Feld DayOfWeek kann in einer Visualisierung verwendet werden, zum Beispiel als Dimension. In einer Tabelle werden die Wochentage automatisch in der richtigen numerischen anstatt der alphabetischen Reihenfolge sortiert.

`Load Dual('Q' & Ceil (Month(Now())/3), Ceil(Month(Now())/3)) as Quarter AutoGenerate 1;`	Dieses Beispiel liefert das aktuelle Quartal. Es wird als Q1 angezeigt, wenn die **Now()-Funktion** in den ersten drei Monaten des Jahres ausgeführt wird, als Q2 für die nächsten drei Monate usw. Wenn es zur Sortierung verwendet wird, verhält sich das Feld Quarter wie ein numerischer Wert: 1 bis 4.
`Dual('Q' & Ceil(Month (Date)/3), Ceil(Month (Date)/3)) as Quarter`	Wie im vorherigen Beispiel wird das Feld Quarter mit den Textwerten 'Q1' bis 'Q4' erstellt und erhält die numerischen Werte 1 bis 4. Damit dies im Skript verwendet werden kann, müssen die Werte für Date geladen werden.
`Dual(WeekYear(Date) & '-W' & Week(Date), WeekStart(Date)) as YearWeek`	Dieses Beispiel erstellt ein YearWeek-Feld mit Textwerten in der Form '2012-W22' und weist gleichzeitig einen Zahlenwert zu, welcher der Datumszahl des ersten Wochentags entspricht, wie zum Beispiel: 41057. Damit dies im Skript verwendet werden kann, müssen die Werte für Date geladen werden.

Interval

Interval() formatiert eine Zahl als Zeitintervall und verwendet dafür das vorgegebene Format aus den Systemvariablen des Datenladeskripts, des Betriebssystems oder eines Formatcodes, wenn dieser zur Verfügung steht.

Intervalle können als Uhrzeiten, Tage oder als Kombination von Tagen, Stunden, Minuten, Sekunden und Sekundenbruchteilen formatiert werden.

Syntax:

```
Interval(number[, format])
```

Rückgabe Datentyp: dual

Argumente:

Argumente

Argument	Beschreibung
number	Die zu formatierende Zahl.
format	String zur Formatierung des resultierenden Zeitintervallstrings. Wenn nicht angegeben, werden das kurze Datumsformat, das Zeitformat und das Dezimaltrennzeichen wie im Betriebssystem festgelegt verwendet.

Beispiele und Ergebnisse:
Bei den nachstehenden Beispielen gehen wir von folgenden Standardeinstellungen aus:

- Datumsformateinstellung 1: YY-MM-DD
- Datumsformateinstellung 2: hh:mm:ss
- Dezimaltrennzeichen: .

Ergebnistabelle

Beispiel	String	Zahl
`Interval( A )` wobei A=0.375	09:00:00	0.375
`Interval( A )` wobei A=1.375	33:00:00	1.375
`Interval( A, 'D hh:mm' )` wobei A=1.375	1 09:00	1.375
`Interval( A-B, 'D hh:mm' )` dabei gilt: A=97-08-06 09:00:00 und B=96-08-06 00:00:00	365 09:00	365.375

Money

Money() formatiert eine Formel numerisch als Geldwert und verwendet dafür, wenn kein Formatcode zur Verfügung steht, das vorgegebene Format aus den Systemvariablen des Datenladeskripts oder des Betriebssystems sowie optionale Dezimal- und Tausendertrennzeichen.

Syntax:

```
Money(number[, format[, dec_sep[, thou_sep]]])
```

Rückgabe Datentyp: dual

Argumente:

Argumente

Argument	Beschreibung
number	Die zu formatierende Zahl.
format	String zur Formatierung des resultierenden Währungsstrings.
dec_sep	String zur Angabe des Dezimaltrennzeichens.
thou_sep	String zur Angabe des Tausendertrennzeichens.

Fehlt das zweite, dritte und vierte Argument, wird das vom Betriebssystem vorgegebene Währungsformat verwendet.

Beispiele und Ergebnisse:
Bei den nachstehenden Beispielen gehen wir von folgenden Standardeinstellungen aus:

- MoneyFormat-Einstellung 1: kr ##0,00, MoneyThousandSep' '
- MoneyFormat-Einstellung 2: $ #,##0.00, MoneyThousandSep','

Beispiel:

```
Money( A )
```
wobei A=35,648

Ergebnistabelle

Results	Standardformat 1	Standardformat 2
String:	€ 35 648,00	$ 35,648.00
Zahl:	35648.00	35648.00

Beispiel:

```
Money( A, '#,##0 ¥', '.' , ',' )
```
wobei A=3,564,800

Ergebnistabelle

Results	Standardformat 1	Standardformat 2
String:	3,564,800 ¥	3,564,800 ¥
Zahl:	3564800	3564800

Num

Num() formatiert eine Zahl, konvertiert also den Zahlenwert der Eingabe in den Eingabetext mit dem Format, das im zweiten Parameter angegeben ist. Wenn der zweite Parameter ausgelassen wird, werden die Dezimal- und Tausendertrennzeichen verwendet, die im Datenladeskript festgelegt sind. Angepasste Dezimal- und Tausendertrennzeichen sind optionale Parameter.

Syntax:

```
Num(number[, format[, dec_sep [, thou_sep]]])
```

Rückgabe Datentyp: dual

Die Funktion Num gibt einen dualen Wert mit dem String- und dem Zahlenwert zurück. Die Funktion nimmt den Zahlenwert der Eingabeformel und generiert einen String, der die Zahl repräsentiert.

Argumente:

Argumente

Argument	Beschreibung
number	Die zu formatierende Zahl.
format	String zur Angabe der Formatierung des resultierenden Strings. Ist kein String angegeben, werden die Dezimal- und Tausendertrennzeichen verwendet, die im Datenladeskript festgelegt sind.
dec_sep	String zur Angabe des Dezimaltrennzeichens. Ist kein String angegeben, wird der vom Datenladeskript vorgegebene Wert der Variablen „DecimalSep“ verwendet.
thou_sep	String zur Angabe des Tausendertrennzeichens. Ist kein String angegeben, wird der vom Datenladeskript vorgegebene Wert der Variablen „ThousandSep“ verwendet.

Beispiel: Diagrammformel

Beispiel:

Die folgende Tabelle zeigt die Ergebnisse, wenn das Feld A 35648.312 entspricht.

Ergebnisse

A	Ergebnis
Num(A)	35648.312 (abhängig von Umgebungsvariablen im Skript)
Num(A, '0.0', '.')	35648.3
Num(A, '0,00', ',')	35648,31
Num(A, '#,##0.0', '.', ',')	35,648.3
Num(A, '# ##0', ',', ' ')	35 648

Beispiel: Ladeskript

Ladeskript

Num kann im Ladeskript zum Formatieren einer Zahl verwendet werden, selbst wenn die Tausender- und Dezimaltrennzeichen bereits im Skript festgelegt sind. Das folgende Ladeskript enthält spezifische Tausender- und Dezimaltrennzeichen, verwendet dann aber *Num* zum Formatieren von Daten auf unterschiedliche Arten.

Erstellen Sie im **Dateneditor** einen neuen Abschnitt, fügen Sie dann das Beispielskript hinzu und führen Sie es aus. Fügen Sie einem Arbeitsblatt in Ihrer App dann mindestens diejenigen Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um das Ergebnis anzuzeigen.

```
SET ThousandSep=',';
SET DecimalSep='.';
Transactions:
Load
*,
Num(transaction_amount) as [No formatting],
Num(transaction_amount,'0') as [0],
Num(transaction_amount,'#,##0') as [#,##0],
Num(transaction_amount,'# ###,00') as [# ###,00],
Num(transaction_amount,'# ###,00',',',' ') as [# ###,00 , ',' , ' '],
Num(transaction_amount,'#,###.00','.',',') as [#,###.00 , '.' , ','],
Num(transaction_amount,'$#,###.00') as [$#,###.00],
;
Load * Inline [
transaction_id, transaction_date, transaction_amount, transaction_quantity, discount,
customer_id, size, color_code
3750, 20180830, 12423.56, 23, 0,2038593, L, Red
3751, 20180907, 5356.31, 6, 0.1, 203521, m, orange
3752, 20180916, 15.75, 1, 0.22, 5646471, S, blue
3753, 20180922, 1251, 7, 0, 3036491, l, Black
3754, 20180922, 21484.21, 1356, 75, 049681, xs, Red
3756, 20180922, -59.18, 2, 0.333333333333333, 2038593, M, Blue
3757, 20180923, 3177.4, 21, .14, 203521, XL, Black
];
```

Qlik Sense Tabelle mit den Ergebnissen verschiedener Verwendungsweisen der *Num*-Funktion im Ladeskript. Die vierte Spalte der Tabelle enthält zu Beispielzwecken eine falsche Verwendung bei der Formatierung.

No formatting	**0**	**#,##0**	**# ###,00**	**# ###,00 , ',' , ' '**	**#,###.00 , '.' , ','**	**$#,###.00**
-59.18	-59	-59	-59###,00	-59,18	-59.18	$-59,18
15.75	16	16	16###,00	15,75	15.75	$15,75
1251	1251	1,251	1251###,00	1 251,00	1,251.00	$1,251.00
3177.4	3177	3,177	3177###,00	3 177,40	3,177.40	$3,177.40
5356.31	5356	5,356	5356###,00	5 356,31	5,356.31	$5,356.31
12423.56	12424	12,424	12424###,00	12 423,56	12,423.56	$12,423.56
21484.21	21484	21,484	21484###,00	21 484,21	21,484.21	$21,484.21

Beispiel: Ladeskript

Ladeskript

Num kann in einem Ladeskript zum Formatieren einer Zahl als Prozentsatz verwendet werden.

Erstellen Sie im **Dateneditor** einen neuen Abschnitt, fügen Sie dann das Beispielskript hinzu und führen Sie es aus. Fügen Sie einem Arbeitsblatt in Ihrer App dann mindestens diejenigen Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um das Ergebnis anzuzeigen.

```
SET ThousandSep=',';
SET DecimalSep='.';
Transactions:
Load
*,
Num(discount,'#,##0%') as [Discount #,##0%]
;
Load * Inline [
transaction_id, transaction_date, transaction_amount, transaction_quantity, discount,
customer_id, size, color_code
3750, 20180830, 12423.56, 23, 0,2038593, L, Red
3751, 20180907, 5356.31, 6, 0.1, 203521, m, orange
3752, 20180916, 15.75, 1, 0.22, 5646471, S, blue
3753, 20180922, 1251, 7, 0, 3036491, l, Black
3754, 20180922, 21484.21, 1356, 75, 049681, xs, Red
3756, 20180922, -59.18, 2, 0.333333333333333, 2038593, M, Blue
3757, 20180923, 3177.4, 21, .14, 203521, XL, Black
];
```

Qlik Sense Tabelle mit den Ergebnissen der *Num*-Funktion, die im Ladeskript zum Formatieren von Prozentsätzen verwendet wird.

Discount	Discount #,##0%
0.333333333333333	33%
0.22	22%
0	0%
.14	14%
0.1	10%
0	0%
75	7,500%

Time

Time() formatiert eine Formel als Zeitwert und verwendet dafür, wenn kein Formatcode zur Verfügung steht, das vorgegebene Format aus den Systemvariablen des Datenladeskripts oder des Betriebssystems.

Syntax:

```
Time(number[, format])
```

Rückgabe Datentyp: dual

Argumente:

Argumente

Argument	Beschreibung
number	Die zu formatierende Zahl.
format	String zur Formatierung des resultierenden Zeitstrings. Wenn nicht angegeben, werden das kurze Datumsformat, das Zeitformat und das Dezimaltrennzeichen wie im Betriebssystem festgelegt verwendet.

Beispiele und Ergebnisse:
Bei den nachstehenden Beispielen gehen wir von folgenden Standardeinstellungen aus:

- Zeitformateinstellung 1: hh:mm:ss
- Zeitformateinstellung 2: hh.mm.ss

Beispiel:

```
Time( A )
```
wobei A=0.375

Ergebnistabelle

Results	Standardformat 1	Standardformat 2
String:	09:00:00	09.00.00
Zahl:	0.375	0.375

Beispiel:

```
Time( A )
```
wobei A=35,648.375

Ergebnistabelle

Results	Standardformat 1	Standardformat 2
String:	09:00:00	09.00.00
Zahl:	35648.375	35648.375

Beispiel:

```
Time( A, 'hh-mm' )
```
wobei A=0.99999

Ergebnistabelle

Results	Standardformat 1	Standardformat 2
String:	23-59	23-59
Zahl:	0.99999	0.99999

Timestamp

TimeStamp() formatiert eine Formel als Datum- und Zeitwert und verwendet dafür, wenn kein Formatcode zur Verfügung steht, das vorgegebene Zeitstempelformat aus den Systemvariablen des Datenladeskripts oder des Betriebssystems.

Syntax:

```
Timestamp(number[, format])
```

Rückgabe Datentyp: dual

Argumente:

Argumente

Argument	Beschreibung
number	Die zu formatierende Zahl.
format	String zur Formatierung des resultierenden Zeitstempelstrings. Wenn nicht angegeben, werden das kurze Datumsformat, das Zeitformat und das Dezimaltrennzeichen wie im Betriebssystem festgelegt verwendet.

Beispiele und Ergebnisse:
Bei den nachstehenden Beispielen gehen wir von folgenden Standardeinstellungen aus:

- TimeStampFormat-Einstellung 1: YY-MM-DD hh:mm:ss
- TimeStampFormat-Einstellung 2: M/D/YY hh:mm:ss

Beispiel:

```
Timestamp( A )
```
wobei A=35,648.375

Ergebnistabelle

Results	Standardformat 1	Standardformat 2
String:	97-08-06 09:00:00	8/6/97 09:00:00
Zahl:	35648.375	35648.375

Beispiel:

```
Timestamp( A,'YYYY-MM-DD hh.mm')
```
wobei A=35,648

Ergebnistabelle

Results	Standardformat 1	Standardformat 2
String:	1997-08-06 00.00	1997-08-06 00.00
Zahl:	35648	35648

8.13 Numerische Funktionen

In diesen numerischen Diagrammfunktionen sind die Argumente Formeln, in denen **x** für eine reelle Zahl steht. Alle Funktionen können sowohl im Datenladeskript als auch in den Diagrammformeln verwendet werden.

Numerische Funktionen – Übersicht

Jede Funktion wird nach der Übersicht genauer beschrieben. Sie können auch auf den Funktionsnamen in der Syntax klicken, um direkt auf die Details zu der spezifischen Funktion zuzugreifen.

bitcount

BitCount() gibt zurück, wie viele Bits in der binären Entsprechung einer Dezimalzahl auf 1 gesetzt sind. Die Funktion liefert also die Zahl der in **integer_number** festgelegten Bits. Dabei wird **integer_number** als ganze Zahl mit Vorzeichen in der 32-Bit-Darstellung interpretiert.

```
BitCount(integer_number)
```

div

Div() liefert den ganzzahligen Teil des arithmetischen Quotienten des ersten und zweiten Arguments. Beide Parameter werden als reelle Zahlen interpretiert, d. h. sie müssen nicht ganzzahlig sein.

```
Div (integer_number1, integer_number2)
```

fabs

Fabs() liefert den Betrag von **x**. Das Ergebnis ist eine positive Zahl.

```
Fabs (x)
```

fact

Fact() liefert die Fakultät der ganzen positiven Zahl **x**.

```
Fact (x)
```

frac

Frac() liefert das Quantil von **x**.

```
Frac (x)
```

sign

Sign() liefert 1, 0, oder -1 je nachdem, ob **x** positiv, 0 oder negativ ist.

```
Sign (x)
```

Kombinations- und Variationsfunktionen

combin

Combin() liefert die Anzahl der möglichen Kombinationen von **q** Elementen aus einer Menge von **p** Objekten. Dies entspricht der folgenden Formel: `Combin(p,q) = p! / q!(p-q)!` Die Reihenfolge, in der die Elemente ausgewählt werden, ist unbedeutend.

```
Combin (p, q)
```

permut

Permut() liefert die Anzahl der möglichen Permutationen von **q** Elementen aus einer Menge von **p**Objekten. Dies entspricht der folgenden Formel: `Permut(p,q) = (p)! / (p - q)!` Die Reihenfolge, in der die Elemente ausgewählt werden, ist signifikant.

```
Permut (p, q)
```

Modulo-Funktionen

fmod

fmod() ist eine generalisierte Modulo-Funktion, die den verbleibenden Teil des ganzzahligen Quotienten des ersten (der Dividend) und zweiten Arguments (der Divisor) liefert. Das Ergebnis ist eine reelle Zahl. Beide Argumente werden als reelle Zahlen interpretiert, d. h. sie müssen nicht ganzzahlig sein.

```
Fmod (a, b)
```

mod

Mod() ist eine mathematische Modulo-Funktion, die den nicht negativen Restwert einer Division mit ganzen Zahlen liefert. Das erste Argument ist der Dividend, das zweite Argument der Divisor, beide Argumente müssen Ganzzahlwerte sein.

```
Mod (integer_number1, integer_number2)
```

Paritätsfunktionen

even

Even() liefert True (-1), wenn **integer_number** eine gerade ganze Zahl oder Null ist. Die Funktion liefert False (0), wenn **integer_number** eine ungerade ganze Zahl ist, und NULL, wenn **integer_number** keine ganze Zahl ist.

```
Even (integer_number)
```

odd

Odd() liefert True (-1), wenn **integer_number** eine ungerade ganze Zahl oder Null ist. Die Funktion liefert False (0), wenn **integer_number** eine gerade ganze Zahl ist, und NULL, wenn **integer_number** keine ganze Zahl ist.

```
Odd (integer_number)
```

Rundungsfunktionen

ceil

Ceil() rundet eine Zahl auf das nächste Vielfache des angegebenen **step**-Intervalls auf, verschoben durch den **offset** -Wert.

```
Ceil (x[, step[, offset]])
```

floor

Floor() rundet eine Zahl auf das nächste Vielfache des angegebenen **step**-Intervalls ab, verschoben durch den **offset** -Wert.

```
Floor (x[, step[, offset]])
```

round
Round() liefert das Ergebnis beim Auf- oder Abrunden auf das nächste Vielfache von **step**, verschoben um **offset** .

```
Round ( x [ , Schritt [ , Offset ]])
```

BitCount

BitCount() gibt zurück, wie viele Bits in der binären Entsprechung einer Dezimalzahl auf 1 gesetzt sind. Die Funktion liefert also die Zahl der in **integer_number** festgelegten Bits. Dabei wird **integer_number** als ganze Zahl mit Vorzeichen in der 32-Bit-Darstellung interpretiert.

Syntax:

```
BitCount(integer_number)
```

Rückgabe Datentyp: ganze Zahl

Beispiele und Ergebnisse:

Beispiele und Ergebnisse

Beispiele	Results
`BitCount ( 3 )`	3 ist binär 11, deshalb wird dadurch 2 zurückgegeben
`BitCount ( -1 )`	-1 entspricht 64-mal der Ziffer eins, wodurch 64 zurückgegeben wird

Ceil

Ceil() rundet eine Zahl auf das nächste Vielfache des angegebenen **step**-Intervalls auf, verschoben durch den **offset** -Wert.

Vergleichbar mit der Funktion **floor**, die Werte abrundet.

Syntax:

```
Ceil(x[, step[, offset]])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
x	Eingabezahl.
step	Intervallschritt. Der Standardwert ist 1.
offset	Definiert die Basis des Schrittintervalls. Der Standardwert ist 0.

Beispiele und Ergebnisse:

Beispiele und Ergebnisse

Beispiele	Ergebnisse
`Ceil(2.4 )`	Liefert 3 In diesem Beispiel ist die Größe des Schritts 1 und die Basis des Schrittintervalls ist 0. Die Intervalle sind ...0 < x <=1, 1 < x <= 2, **2< x <=3**, 3< x <=4....
`Ceil(4.2 )`	Liefert 5
`Ceil(3.88 ,0.1)`	Liefert 3,9 In diesem Beispiel ist die Größe des Schritts 0,1 und die Basis des Intervalls ist 0. Die Intervalle sind ... 3.7 < x <= 3.8, **3.8 < x <= 3.9**, 3.9 < x <= 4.0....
`Ceil(3.88 ,5)`	Liefert 5
`Ceil(1.1 ,1)`	Liefert 2
`Ceil(1.1 ,1,0.5)`	Liefert 1,5 In diesem Beispiel ist die Größe des Schritts 1 und die Basis des Schrittintervalls ist 0,5. Das bedeutet, dass die Basis des Schrittintervalls 0,5 ist und nicht 0. Die Intervalle sind ...**0.5 < x <=1.5**, 1.5 < x <= 2.5, 2.5< x <=3.5, 3.5< x <=4.5....
`Ceil(1.1 ,1,-0.01)`	Liefert 1,99 Die Intervalle sind ...-0.01< x <= 0.99, **0.99< x <= 1.99**, 1.99 < x <=2.99....

Combin

Combin() liefert die Anzahl der möglichen Kombinationen von **q** Elementen aus einer Menge von **p** Objekten. Dies entspricht der folgenden Formel: `Combin(p,q) = p! / q!(p-q)!` Die Reihenfolge, in der die Elemente ausgewählt werden, ist unbedeutend.

Syntax:

```
Combin(p, q)
```

Rückgabe Datentyp: ganze Zahl

Beschränkungen:

Bei Dezimalzahlen werden die Nachkommastellen ignoriert.

Beispiele und Ergebnisse:

Beispiele und Ergebnisse

Beispiele	Ergebnisse
Wie viele Kombinationen sind beim Lottospiel möglich, wenn 7 aus 35 Kugeln gezogen werden? `Combin( 35,7 )`	Liefert 6.724.520

Div

Div() liefert den ganzzahligen Teil des arithmetischen Quotienten des ersten und zweiten Arguments. Beide Parameter werden als reelle Zahlen interpretiert, d. h. sie müssen nicht ganzzahlig sein.

Syntax:

```
Div(integer_number1, integer_number2)
```

Rückgabe Datentyp: ganze Zahl

Beispiele und Ergebnisse:

Beispiele und Ergebnisse

Beispiele	Ergebnisse
`Div( 7,2 )`	Liefert 3
`Div( 7.1,2.3 )`	Liefert 3
`Div( 9,3 )`	Liefert 3
`Div( -4,3 )`	Liefert -1
`Div( 4,-3 )`	Liefert -1
`Div( -4,-3 )`	Liefert 1

Even

Even() liefert True (-1), wenn **integer_number** eine gerade ganze Zahl oder Null ist. Die Funktion liefert False (0), wenn **integer_number** eine ungerade ganze Zahl ist, und NULL, wenn **integer_number** keine ganze Zahl ist.

Syntax:

```
Even(integer_number)
```

Rückgabe Datentyp: Boolesch

Beispiele und Ergebnisse:

Beispiele und Ergebnisse

Beispiele	Ergebnisse
`Even( 3 )`	Liefert 0, False
`Even( 2 * 10 )`	Liefert -1, True
`Even( 3.14 )`	Liefert NULL

Fabs

Fabs() liefert den Betrag von **x**. Das Ergebnis ist eine positive Zahl.

Syntax:

```
fabs(x)
```

Rückgabe Datentyp: numerisch

Beispiele und Ergebnisse:

Beispiele und Ergebnisse

Beispiele	Ergebnisse
`fabs( 2.4 )`	Liefert 2,4
`fabs( -3.8 )`	Liefert 3,8

Fact

Fact() liefert die Fakultät der ganzen positiven Zahl **x**.

Syntax:

```
Fact(x)
```

Rückgabe Datentyp: ganze Zahl

Beschränkungen:

Ist die Zahl **x** keine ganze Zahl, bleiben die Nachkommastellen unberücksichtigt. Bei negativen Zahlen ist das Ergebnis NULL.

Beispiele und Ergebnisse:

Beispiele und Ergebnisse

Beispiele	Ergebnisse
`Fact( 1 )`	Liefert 1
`Fact( 5 )`	Liefert 120 (1 * 2 * 3 * 4 * 5 = 120)
`Fact( -5 )`	Liefert NULL

Floor

Floor() rundet eine Zahl auf das nächste Vielfache des angegebenen **step**-Intervalls ab, verschoben durch den **offset** -Wert.

Vergleichbar mit der Funktion **ceil**, die Werte aufrundet.

Syntax:

```
Floor(x[, step[, offset]])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
x	Eingabezahl.
step	Intervallschritt. Der Standardwert ist 1.
offset	Definiert die Basis des Schrittintervalls. Der Standardwert ist 0.

Beispiele und Ergebnisse:

Beispiele und Ergebnisse

Beispiele	Ergebnisse
`Floor(2.4)`	Liefert 2 In this example, the size of the step is 1 and the base of the step interval is 0. The intervals are ...0 <= x <1, 1 <= x < 2, **2<= x <3**, 3<= x <4....
`Floor(4.2)`	Liefert 4

Beispiele	Ergebnisse
`Floor(3.88 ,0.1)`	Liefert 3,8 In diesem Beispiel ist die Größe des Schritts 0,1 und die Basis des Intervalls ist 0. Die Intervalle sind ... 3.7 <= x < 3.8, **3.8 <= x < 3.9**, 3.9 <= x < 4.0....
`Floor(3.88 ,5)`	Gibt 0 zurück.
`Floor(1.1 ,1)`	Liefert 1
`Floor(1.1 ,1,0.5)`	Liefert 0,5 In diesem Beispiel ist die Größe des Schritts 1 und die Differenz ist 0,5. Das bedeutet, dass die Basis des Schrittintervalls 0,5 ist und nicht 0. Die Intervalle sind ...**0.5 <= x <1.5**, 1.5 <= x < 2.5, 2.5<= x <3.5,....

Fmod

fmod() ist eine generalisierte Modulo-Funktion, die den verbleibenden Teil des ganzzahligen Quotienten des ersten (der Dividend) und zweiten Arguments (der Divisor) liefert. Das Ergebnis ist eine reelle Zahl. Beide Argumente werden als reelle Zahlen interpretiert, d. h. sie müssen nicht ganzzahlig sein.

Syntax:

```
fmod(a, b)
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
a	Dividende
b	Divisor

Beispiele und Ergebnisse:

Beispiele und Ergebnisse

Beispiele	Ergebnisse
`fmod( 7,2 )`	Liefert 1
`fmod( 7.5,2 )`	Liefert 1,5
`fmod( 9,3 )`	Liefert 0
`fmod( -4,3 )`	Liefert -1
`fmod( 4,-3 )`	Liefert 1
`fmod( -4,-3 )`	Liefert -1

Frac

Frac() liefert das Quantil von **x**.

Der Bruchteil wird wie folgt definiert: `Frac(x ) + Floor(x ) = x`. Das heißt also, dass der Bruchteil einer positiven Zahl der Unterschied zwischen der Zahl (x) und der vor dem Bruchteil stehenden ganzen Zahl ist.

Hier ein Beispiel: Der Bruchteil von 11,43 = 11,43 - 11 = 0,43

Bei einer negativen Zahl wie -1,4 ist Floor(-1.4) = -2, was zu folgendem Ergebnis führt:

Der Bruchteil von -1,4 = 1,4 - (-2) = -1,4 + 2 = 0,6

Syntax:

```
Frac(x)
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
x	Zahl für die Rückgabe des Bruchteils.

Beispiele und Ergebnisse:

Beispiele und Ergebnisse

Beispiele	Ergebnisse
`Frac( 11.43 )`	Liefert 0,43
`Frac( -1.4 )`	Liefert 0,6
Extrahiert die Zeitkomponente aus der Zahlendarstellung eines Zeitstempels und lässt somit das Datum aus. `Time(Frac(44518.663888889))`	Gibt 3:56:00 PM zurück

Mod

Mod() ist eine mathematische Modulo-Funktion, die den nicht negativen Restwert einer Division mit ganzen Zahlen liefert. Das erste Argument ist der Dividend, das zweite Argument der Divisor, beide Argumente müssen Ganzzahlwerte sein.

Syntax:

```
Mod(integer_number1, integer_number2)
```

Rückgabe Datentyp: ganze Zahl

Beschränkungen:

Ferner muss **integer_number2** größer als 0 sein.

Beispiele und Ergebnisse:

Beispiele und Ergebnisse

Beispiele	Ergebnisse
`Mod( 7,2 )`	Liefert 1
`Mod( 7.5,2 )`	Liefert NULL
`Mod( 9,3 )`	Liefert 0
`Mod( -4,3 )`	Liefert 2
`Mod( 4,-3 )`	Liefert NULL
`Mod( -4,-3 )`	Liefert NULL

Odd

Odd() liefert True (-1), wenn **integer_number** eine ungerade ganze Zahl oder Null ist. Die Funktion liefert False (0), wenn **integer_number** eine gerade ganze Zahl ist, und NULL, wenn **integer_number** keine ganze Zahl ist.

Syntax:

```
Odd(integer_number)
```

Rückgabe Datentyp: Boolesch

Beispiele und Ergebnisse:

Beispiele und Ergebnisse

Beispiele	Ergebnisse
`Odd( 3 )`	Liefert -1, True
`Odd( 2 * 10 )`	Liefert 0, False
`Odd( 3.14 )`	Liefert NULL

Permut

Permut() liefert die Anzahl der möglichen Permutationen von **q** Elementen aus einer Menge von **p**Objekten. Dies entspricht der folgenden Formel: `Permut(p,q) = (p)! / (p - q)!` Die Reihenfolge, in der die Elemente ausgewählt werden, ist signifikant.

Syntax:

```
Permut(p, q)
```

Rückgabe Datentyp: ganze Zahl

Beschränkungen:

Bei Dezimalzahlen werden die Nachkommastellen ignoriert.

Beispiele und Ergebnisse:

Beispiele und Ergebnisse

Beispiele	Ergebnisse
Wie viele verschiedene Möglichkeiten gibt es, die Gold-, Silber- und Bronzemedaillen nach einem 100-m-Finale in einer Gruppe von 8 Läufern zu verteilen? `Permut( 8,3 )`	Liefert 336

Round

Round() liefert das Ergebnis beim Auf- oder Abrunden auf das nächste Vielfache von **step**, verschoben um **offset** .

Liegt die Zahl genau in der Mitte eines Intervalls, wird aufgerundet.

Syntax:

```
Round(x[, step[, offset]])
```

Rückgabe Datentyp: numerisch

> *Wenn Sie Gleitkommazahlen auf- oder abrunden, sind die Ergebnisse möglicherweise fehlerhaft. Diese Rundungsfehler sind darauf zurückzuführen, dass Gleitkommazahlen durch eine endliche Anzahl von Binärstellen dargestellt werden. Deshalb werden die Ergebnisse mit einer bereits auf- oder abgerundeten Zahl berechnet. Wenn Rundungsfehler sich auf Ihre Arbeit auswirken, können die Zahlen multipliziert werden, um sie vor der Auf-/Abrundung in ganze Zahlen umzuwandeln.*

Argumente:

Argumente

Argument	Beschreibung
x	Eingabezahl.
step	Intervallschritt. Der Standardwert ist 1.
offset	Definiert die Basis des Schrittintervalls. Der Standardwert ist 0.

Beispiele und Ergebnisse:

Beispiele und Ergebnisse

Beispiele	Ergebnisse
`Round(3.8 )`	Liefert 4 In diesem Beispiel ist die Größe des Schritts 1 und die Basis des Schrittintervalls ist 0. Die Intervalle sind ...0 <= x <1, 1 <= x < 2, 2<= x <3, **3<= x <4**....
`Round(3.8,4 )`	Liefert 4
`Round(2.5 )`	Liefert 3. In diesem Beispiel ist die Größe des Schritts 1 und die Basis des Schrittintervalls ist 0. Die Intervalle sind ...0 <= x <1, 1 <= x <2, **2<= x <3**...
`Round(2,4 )`	Gibt 4 zurück. Aufgerundet, weil 2 genau der Hälfte des Schrittintervalls von 4 entspricht. In diesem Beispiel ist die Größe des Schritts 4 und die Basis des Schrittintervalls 0. Die Intervalle sind ...**0 <= x <4**, 4 <= x <8, 8<= x <12...
`Round(2,6 )`	Gibt 0 zurück. Abgerundet, weil 2 weniger als der Hälfte des Schrittintervalls von 6 entspricht. In diesem Beispiel ist die Größe des Schritts 6 und die Basis des Schrittintervalls ist 0. Die Intervalle sind ...**0 <= x <6**, 6 <= x <12, 12<= x <18...
`Round(3.88 ,0.1)`	Liefert 3,9 In diesem Beispiel ist die Größe des Schritts 0,1 und die Basis des Schrittintervalls ist 0. Die Intervalle sind ... 3.7 <= x <3.8, **3.8 <= x <3.9**, 3.9 <= x < 4.0....
`Round (3.88875,1/1000)`	Gibt 3.889 zurück In diesem Beispiel ist die Größe des Schritts 0,001, wodurch die Zahl aufgerundet und auf drei Dezimalstellen beschränkt wird.
`Round(3.88 ,5)`	Liefert 5
`Round(1.1 ,1,0.5)`	Liefert 1,5 In diesem Beispiel ist die Größe des Schritts 1 und die Basis des Schrittintervalls ist 0,5. Die Intervalle sind ...**0.5 <= x <1.5**, 1.5 <= x <2.5, 2.5<= x <3.5....

Sign

Sign() liefert 1, 0, oder -1 je nachdem, ob **x** positiv, 0 oder negativ ist.

Syntax:

```
Sign(x)
```

Rückgabe Datentyp: numerisch

Beschränkungen:

Werden keine numerischen Werte gefunden, ist das Ergebnis NULL.

Beispiele und Ergebnisse:

Beispiele und Ergebnisse

Beispiele	Ergebnisse
`Sign( 66 )`	Liefert 1
`Sign( 0 )`	Liefert 0
`Sign( - 234 )`	Liefert -1

8.14 Räumliche Funktionen

Diese Funktionen dienen zur Bearbeitung von räumlichen Daten in Kartenvisualisierungen. Qlik Sense folgt den GeoJSON-Spezifikationen für räumliche Daten und unterstützt folgende Optionen:

- Punkt
- Linestring
- Polygon
- Multipolygon

Weitere Informationen zu GeoJSON-Spezifikationen finden Sie unter:
GeoJSON.org

Räumliche Funktionen – Übersicht

Jede Funktion wird nach der Übersicht genauer beschrieben. Sie können auch auf den Funktionsnamen in der Syntax klicken, um direkt auf die Details zu der spezifischen Funktion zuzugreifen.

Es gibt zwei Kategorien räumlicher Funktionen: Aggregierungs- und Nichtaggregierungsfunktionen.

Aggregierungsfunktionen nutzen einen Geometriesatz (Punkte oder Bereiche) als Eingangsgröße und geben eine einzelne Geometriegröße aus. So können beispielsweise mehrere Bereiche verbunden werden und eine einzige Grenze kann für die Aggregierung auf der Karte gezogen werden.

Funktionen ohne Aggregierung nutzen eine einzelne Geometriegröße und geben eine einzige zurück. Wird beispielsweise bei der Funktion GeoGetPolygonCenter() die Grenzgeometrie eines Bereichs als Eingangsgröße festgelegt, wird die Punktgeometrie (Längen- und Breitengrad) für die Mitte dieses Bereichs zurückgegeben.

Nachfolgende Funktionen sind Aggregierungsfunktionen:

GeoAggrGeometry
GeoAggrGeometry() kann dazu verwendet werden, mehrere Gebiete in ein größeres Gebiet zu aggregieren, beispielsweise mehrere Untergebiete in eine Region.

```
GeoAggrGeometry (field_name)
```

GeoBoundingBox
GeoBoundingBox() kann verwendet werden, um eine Geometrie in ein Gebiet zu aggregieren und die kleinste Grenzbox zu berechnen, die alle Koordinaten enthält.

```
GeoBoundingBox (field_name)
```

GeoCountVertex
GeoCountVertex() kann dazu verwendet werden, die Anzahl der Eckpunkte zu ermitteln, die eine Polygongeometrie enthält.

```
GeoCountVertex(field_name)
```

GeoInvProjectGeometry
GeoInvProjectGeometry() kann verwendet werden, um eine Geometrie in ein Gebiet zu aggregieren und die Umkehrung einer Projektion anzuwenden.

```
GeoInvProjectGeometry(type, field_name)
```

GeoProjectGeometry
GeoProjectGeometry() kann verwendet werden, um eine Geometrie in ein Gebiet zu aggregieren und eine Projektion anzuwenden.

```
GeoProjectGeometry(type, field_name)
```

GeoReduceGeometry
GeoReduceGeometry() kann verwendet werden, um die Anzahl der Eckpunkte einer Geometrie zu reduzieren und die Anzahl der Gebiete zu einem Gebiet zu aggregieren, wobei jedoch weiterhin die Grenzlinien der einzelnen Gebiete angezeigt werden.

```
GeoReduceGeometry (geometry)
```

Nachfolgende Funktionen sind Funktionen ohne Aggregierung:

GeoGetBoundingBox
GeoGetBoundingBox() kann in Skripten und Diagrammformeln zum Berechnen der kleinsten räumlichen Grenzbox verwendet werden, die alle Koordinaten einer Geometrie enthält.

```
GeoGetBoundingBox (geometry)
```

GeoGetPolygonCenter

GeoGetPolygonCenter() kann in Skripten und Diagrammformeln zum Berechnen und Angeben des Mittelpunkts einer Geometrie verwendet werden.

```
GeoGetPolygonCenter (geometry)
```

GeoMakePoint

GeoMakePoint() kann in Skripten und Diagrammformeln zum Erstellen und Markieren eines Punkts mit Längen- und Breitengrad verwendet werden.

```
GeoMakePoint (lat_field_name, lon_field_name)
```

GeoProject

GeoProject() kann in Skripten und Diagrammformeln zum Anwenden einer Projektion auf eine Geometrie verwendet werden.

```
GeoProject (type, field_name)
```

GeoAggrGeometry

GeoAggrGeometry() kann dazu verwendet werden, mehrere Gebiete in ein größeres Gebiet zu aggregieren, beispielsweise mehrere Untergebiete in eine Region.

Syntax:

```
GeoAggrGeometry(field_name)
```

Rückgabe Datentyp: String

Argumente:

Argumente

Argument	Beschreibung
field_name	Ein Feld oder eine Formel, die sich auf ein Feld bezieht, das die zu definierende Geometrie enthält. Hierbei kann es sich um einen Punkt (oder einen Punktesatz) handeln, der Längen- und Breitengrad bezeichnet, oder um ein Gebiet.

Normalerweise können räumliche Grenzdaten mithilfe von **GeoAggrGeometry()** kombiniert werden. So werden möglicherweise Postleitzahlbereiche für Vororte einer Stadt und Verkaufsumsätze für jedes einzelne Gebiet verwendet. Deckt das Territorium einer Verkaufsperson mehrere Postleitzahlbereiche ab, ist es vielleicht nützlich, den Gesamtumsatz nach Verkaufsterritorium anstatt nach den einzelnen Bereichen anzuzeigen und diese Ergebnisse auf einer farbigen Karte darstellen zu lassen.

GeoAggrGeometry() kann die Aggregierung der einzelnen Geometriegrößen für die Vororte berechnen und eine verbundene Territoriumsgeometrie im Datenmodell erstellen. Falls später die Anpassung der Grenzen des Verkaufsterritoriums erfolgt, werden die neuen verbundenen Grenzen und Umsätze beim erneuten Laden der Daten auf der Karte angezeigt.

Da es sich bei **GeoAggrGeometry()** um eine aggregierende Funktion handelt, benötigen Sie den Befehl **LOAD** mit einer Bedingung **Group by**, wenn Sie sie in einem Skript verwenden möchten.

*Die Grenzlinien der mithilfe von **GeoAggrGeometry()** erstellten Karten entsprechen denen der verbundenen Bereiche. Falls Sie die einzelnen Grenzlinien der Bereiche vor der Aggregierung anzeigen möchten, verwenden Sie dazu **GeoReduceGeometry()**.*

Beispiele:
Dieses Beispiel lädt zunächst eine KML-Datei mit Bereichsdaten und anschließend eine Tabelle mit den aggregierten Bereichsdaten.

```
[MapSource]: LOAD [world.Name], [world.Point], [world.Area] FROM [lib://Downloads/world.kml]
(kml, Table is [World.shp/Features]); Map: LOAD world.Name, GeoAggrGeometry(world.Area) as
[AggrArea] resident MapSource Group By world.Name;

 Drop Table MapSource;
```

GeoBoundingBox

GeoBoundingBox() kann verwendet werden, um eine Geometrie in ein Gebiet zu aggregieren und die kleinste Grenzbox zu berechnen, die alle Koordinaten enthält.

Eine GeoBoundingBox wird als Liste von vier Werten, links, rechts, oben und unten, dargestellt.

Syntax:

```
GeoBoundingBox(field_name)
```

Rückgabe Datentyp: String

Argumente:

Argumente

Argument	Beschreibung
field_name	Ein Feld oder eine Formel, die sich auf ein Feld bezieht, das die zu definierende Geometrie enthält. Hierbei kann es sich um einen Punkt (oder einen Punktesatz) handeln, der Längen- und Breitengrad bezeichnet, oder um ein Gebiet.

GeoBoundingBox() aggregiert einen Satz von Geometriegrößen und gibt vier Koordinaten für das kleinste Rechteck zurück, das die Koordinaten dieser aggregierten Geometrie enthält.

Zur Visualisierung des Ergebnisses auf einer Karte übertragen Sie den entstehenden String aus vier Koordinaten in ein Polygonformat, kennzeichnen Sie das übertragene Feld mit einem Geopolygonformat und ziehen Sie dieses Feld auf ein Kartenobjekt. Die rechteckigen Felder werden anschließend in der Kartenvisualisierung angezeigt.

GeoCountVertex

GeoCountVertex() kann dazu verwendet werden, die Anzahl der Eckpunkte zu ermitteln, die eine Polygongeometrie enthält.

Syntax:

```
GeoCountVertex(field_name)
```

Rückgabe Datentyp: ganze Zahl

Argumente:

Argumente

Argument	Beschreibung
field_name	Ein Feld oder eine Formel, die sich auf ein Feld bezieht, das die zu definierende Geometrie enthält. Hierbei kann es sich um einen Punkt (oder einen Punktesatz) handeln, der Längen- und Breitengrad bezeichnet, oder um ein Gebiet.

GeoGetBoundingBox

GeoGetBoundingBox() kann in Skripten und Diagrammformeln zum Berechnen der kleinsten räumlichen Grenzbox verwendet werden, die alle Koordinaten einer Geometrie enthält.

Eine räumliche Grenzbox, erstellt durch die Funktion GeoBoundingBox(), wird als Liste von vier Werten dargestellt: links, rechts, oben und unten.

Syntax:

```
GeoGetBoundingBox(field_name)
```

Rückgabe Datentyp: String

Argumente:

Argumente

Argument	Beschreibung
field_name	Ein Feld oder eine Formel, die sich auf ein Feld bezieht, das die zu definierende Geometrie enthält. Hierbei kann es sich um einen Punkt (oder einen Punktesatz) handeln, der Längen- und Breitengrad bezeichnet, oder um ein Gebiet.

*Verwenden Sie nicht die Bedingung **Group by** im Dateneditor mit dieser und anderen räumlichen Nichtaggregierungsfunktionen, weil dadurch beim Laden ein Fehler verursacht wird.*

GeoGetPolygonCenter

GeoGetPolygonCenter() kann in Skripten und Diagrammformeln zum Berechnen und Angeben des Mittelpunkts einer Geometrie verwendet werden.

In manchen Fällen soll ein Punkt anstelle einer Füllfarbe auf einer Karte gedruckt werden. Sind die vorhandenen räumlichen Daten nur in der Form einer Bereichsgeometriegröße verfügbar (zum Beispiel eine Grenze), können Sie **GeoGetPolygonCenter()** zum Abrufen von Längen- und Breitengrad der Mitte des Bereichs verwenden.

Syntax:

```
GeoGetPolygonCenter(field_name)
```

Rückgabe Datentyp: String

Argumente:

Argumente

Argument	Beschreibung
field_name	Ein Feld oder eine Formel, die sich auf ein Feld bezieht, das die zu definierende Geometrie enthält. Hierbei kann es sich um einen Punkt (oder einen Punktesatz) handeln, der Längen- und Breitengrad bezeichnet, oder um ein Gebiet.

*Verwenden Sie nicht die Bedingung **Group by** im Dateneditor mit dieser und anderen räumlichen Nichtaggregierungsfunktionen, weil dadurch beim Laden ein Fehler verursacht wird.*

GeoInvProjectGeometry

GeoInvProjectGeometry() kann verwendet werden, um eine Geometrie in ein Gebiet zu aggregieren und die Umkehrung einer Projektion anzuwenden.

Syntax:

```
GeoInvProjectGeometry(type, field_name)
```

Rückgabe Datentyp: String

Argumente:

Argumente

Argument	Beschreibung
type	Projektionstyp, der für die Umwandlung der Kartengeometrie verwendet wird. Hierbei kann es sich um einen von zwei Werten handeln: 'unit' (Standard), was zu einer 1:1-Projektion führt, oder 'mercator', wodurch die Standard-Mercator-Projektion verwendet wird.

Argument	Beschreibung
field_name	Ein Feld oder eine Formel, die sich auf ein Feld bezieht, das die zu definierende Geometrie enthält. Hierbei kann es sich um einen Punkt (oder einen Punktesatz) handeln, der Längen- und Breitengrad bezeichnet, oder um ein Gebiet.

Beispiel:

Skriptbeispiel

Beispiel	Ergebnis
In einem `Load`-Befehl: `GeoInvProjectGeometry ('mercator',AreaPolygon) as InvProjectGeometry`	Die als **AreaPolygon** geladene Geometrie wird mithilfe der umgekehrten Umformung der Mercator-Projektion umgewandelt und als **InvProjectGeometry** für den Einsatz in Visualisierungen gespeichert.

GeoMakePoint

GeoMakePoint() kann in Skripten und Diagrammformeln zum Erstellen und Markieren eines Punkts mit Längen- und Breitengrad verwendet werden. GeoMakePoint gibt Punkte nach Längengrad und Breitengrad zurück.

Syntax:

```
GeoMakePoint(lat_field_name, lon_field_name)
```

Rückgabe Datentyp: String, formatiert [Längengrad, Breitengrad]

Argumente:

Argumente

Argument	Beschreibung
lat_field_name	Ein Feld oder eine Formel, die sich auf ein Feld bezieht, das den Breitengrad des Punkts bezeichnet.
lon_field_name	Ein Feld oder eine Formel, die sich auf ein Feld bezieht, das den Längengrad des Punkts bezeichnet.

*Verwenden Sie nicht die Bedingung **Group by** im Dateneditor mit dieser und anderen räumlichen Nichtaggregierungsfunktionen, weil dadurch beim Laden ein Fehler verursacht wird.*

GeoProject

GeoProject() kann in Skripten und Diagrammformeln zum Anwenden einer Projektion auf eine Geometrie verwendet werden.

Syntax:

```
GeoProject(type, field_name)
```

Rückgabe Datentyp: String

Argumente:

Argumente

Argument	Beschreibung
type	Projektionstyp, der für die Umwandlung der Kartengeometrie verwendet wird. Hierbei kann es sich um einen von zwei Werten handeln: 'unit' (Standard), was zu einer 1:1-Projektion führt, oder 'mercator', wodurch die Web-Mercator-Projektion verwendet wird.
field_name	Ein Feld oder eine Formel, die sich auf ein Feld bezieht, das die zu definierende Geometrie enthält. Hierbei kann es sich um einen Punkt (oder einen Punktesatz) handeln, der Längen- und Breitengrad bezeichnet, oder um ein Gebiet.

*Verwenden Sie nicht die Bedingung **Group by** im Dateneditor mit dieser und anderen räumlichen Nichtaggregierungsfunktionen, weil dadurch beim Laden ein Fehler verursacht wird.*

Beispiel:

Skriptbeispiele

Beispiel	Ergebnis
In einem `Load`-Befehl: `GeoProject ('mercator',Area) as GetProject`	Die Mercator-Projektion wird auf die geladene Geometrie als **Area** angewandt und das Ergebnis wird als **GetProject** gespeichert.

GeoProjectGeometry

GeoProjectGeometry() kann verwendet werden, um eine Geometrie in ein Gebiet zu aggregieren und eine Projektion anzuwenden.

Syntax:

```
GeoProjectGeometry(type, field_name)
```

Rückgabe Datentyp: String

Argumente:

Argumente

Argument	Beschreibung
type	Projektionstyp, der für die Umwandlung der Kartengeometrie verwendet wird. Hierbei kann es sich um einen von zwei Werten handeln: 'unit' (Standard), was zu einer 1:1-Projektion führt, oder 'mercator', wodurch die Web-Mercator-Projektion verwendet wird.

Argument	Beschreibung
field_name	Ein Feld oder eine Formel, die sich auf ein Feld bezieht, das die zu definierende Geometrie enthält. Hierbei kann es sich um einen Punkt (oder einen Punktesatz) handeln, der Längen- und Breitengrad bezeichnet, oder um ein Gebiet.

Beispiel:

Beispiel	Ergebnis
In einem Load-Befehl: `GeoProjectGeometry ('mercator',AreaPolygon) as ProjectGeometry`	Die als **AreaPolygon** geladene Geometrie wird mithilfe der Mercator-Projektion umgewandelt und als **ProjectGeometry** für den Einsatz in Visualisierungen gespeichert.

GeoReduceGeometry

GeoReduceGeometry() kann verwendet werden, um die Anzahl der Eckpunkte einer Geometrie zu reduzieren und die Anzahl der Gebiete zu einem Gebiet zu aggregieren, wobei jedoch weiterhin die Grenzlinien der einzelnen Gebiete angezeigt werden.

Syntax:

```
GeoReduceGeometry(field_name[, value])
```

Rückgabe Datentyp: String

Argumente:

Argumente

Argument	Beschreibung
field_name	Ein Feld oder eine Formel, die sich auf ein Feld bezieht, das die zu definierende Geometrie enthält. Hierbei kann es sich um einen Punkt (oder einen Punktesatz) handeln, der Längen- und Breitengrad bezeichnet, oder um ein Gebiet.
value	Die auf die Geometrie anzuwendende Reduzierung. Der Bereich liegt zwischen 0 und 1, wobei 0 keine Reduzierung bedeutet und 1 für die maximale Reduzierung der Eckpunkte steht. *Die Verwendung eines value von 0,9 oder höher mit komplexen Daten kann die Anzahl der Eckpunkte soweit reduzieren, dass die visuelle Darstellung ungenau ist.*

GeoReduceGeometry() führt ebenfalls eine vergleichbare Funktion wie**GeoAggrGeometry()** aus, wobei die Anzahl der Gebiete in ein Gebiet aggregiert wird. Der Unterschied zwischen den einzelnen Grenzlinien aus den Daten vor der Aggregierung wird auf der Karte angezeigt, wenn Sie **GeoReduceGeometry()** verwenden.

Da es sich bei **GeoReduceGeometry()** um eine aggregierende Funktion handelt, benötigen Sie den Befehl **LOAD** mit einer Bedingung **Group by**, wenn Sie sie in einem Skript verwenden möchten.

Beispiele:
Dieses Beispiel lädt zunächst eine KML-Datei mit Bereichsdaten und anschließend eine Tabelle mit den reduzierten und aggregierten Bereichsdaten.

```
[MapSource]:
LOAD [world.Name],
       [world.Point],
       [world.Area]
 FROM [lib://Downloads/world.kml]
(kml, Table is [World.shp/Features]);

Map:
LOAD world.Name,
     GeoReduceGeometry(world.Area,0.5) as [ReducedArea]
resident MapSource Group By world.Name;

Drop Table MapSource;
```

8.15 Interpretationsfunktionen

Die Interpretationsfunktionen werten den Inhalt von Eingabetextfeldern oder Formeln aus und wenden ein bestimmtes Datenformat auf den resultierenden numerischen Wert an. Mit Hilfe dieser Funktionen können Sie das Format der Zahl gemäß ihrem Datentyp festlegen, einschließlich Attribute wie: Dezimaltrennzeichen, Tausendertrennzeichen und Datumsformat.

Die Interpretationsfunktionen geben alle einen zweifachen Wert mit dem String und dem Zahlenwert zurück. Die Funktionsweise entspricht aber einer Umwandlung von Strings in Zahlen. Die Funktionen nehmen den Textwert der Eingabeformel und generieren eine Zahl, die den String repräsentiert.

Die Formatfunktionen gehen genau andersherum vor: sie werten numerische Formeln als Strings aus und geben das Anzeigeformat des resultierenden Textes an.

Werden keine Interpretationsfunktionen benutzt, interpretiert Qlik Sense die Daten als Mischung von Zahlen, Daten, Uhrzeiten, Zeitstempeln und Strings, wobei die Systemeinstellungen bzw. die durch die Skriptvariablen definierten Formate verwendet werden.

Alle Interpretationsfunktionen können sowohl im Datenladeskript als auch in den Diagrammformeln verwendet werden.

Bei der Zahlendarstellung wird immer ein Punkt als Dezimaltrennzeichen verwendet.

Interpretationsfunktionen – Übersicht

Jede Funktion wird nach der Übersicht genauer beschrieben. Sie können auch auf den Funktionsnamen in der Syntax klicken, um direkt auf die Details zu der spezifischen Funktion zuzugreifen.

Date#

Date# evaluiert eine Formel als Datum und verwendet dafür das Format, das im zweiten Argument angegeben ist, wenn dies zur Verfügung steht. Ist kein Formatcode angegeben, wird das vom Betriebssystem vorgegebene Datumsformat verwendet.

```
Date# (page 1293)(text[, format])
```

Interval#

Interval#() evaluiert eine Textformel als Zeitintervall und verwendet dafür das standardmäßig vorgegebene Format des Betriebssystems oder das Format, das im zweiten Argument angegeben ist, wenn dies zur Verfügung steht.

```
Interval# (page 1294)(text[, format])
```

Money#

Money#() wandelt einen Textstring in einen Geldwert um und verwendet dafür, wenn kein Formatcode zur Verfügung steht, das vorgegebene Format des Ladeskripts oder des Betriebssystems. Angepasste Dezimal- und Tausendertrennzeichen sind optionale Parameter.

```
Money# (page 1294)(text[, format[, dec_sep[, thou_sep ] ] ])
```

Num#

Num#() interpretiert einen Textstring als Zahlenwert, konvertiert also den Eingabestring in eine Zahl mit dem Format, das im zweiten Parameter angegeben ist. Wenn der zweite Parameter ausgelassen wird, werden die Dezimal- und Tausendertrennzeichen verwendet, die im Datenladeskript festgelegt sind. Angepasste Dezimal- und Tausendertrennzeichen sind optionale Parameter.

```
Num# (page 1296)(text[ , format[, dec_sep[ , thou_sep]]])
```

Text

Text() interpretiert die Formel als Text, auch wenn eine numerische Interpretation möglich ist.

```
Text(expr)
```

Time#

Time#() evaluiert eine Formel als Zeitwert und verwendet dafür, wenn kein Formatcode zur Verfügung steht, das vorgegebene Zeitformat des Datenladeskripts oder des Betriebssystems..

```
Time# (page 1297)(text[, format])
```

Timestamp#

Timestamp#() evaluiert eine Formel als Daten- und Zeitwert und verwendet dafür, wenn kein Formatcode zur Verfügung steht, das vorgegebene Zeitstempelformat des Datenladeskripts oder des Betriebssystems.

```
Timestamp# (page 1298)(text[, format])
```

Siehe auch:

- *Formatfunktionen (page 1256)*

Date#

Date# evaluiert eine Formel als Datum und verwendet dafür das Format, das im zweiten Argument angegeben ist, wenn dies zur Verfügung steht.

Syntax:

```
Date#(text[, format])
```

Rückgabe Datentyp: dual

Argumente:

Argumente

Argument	Beschreibung
text	Der zu evaluierende Textstring.
format	String, der das Format des auszuwertenden Textstrings beschreibt. Ist kein String angegeben, wird das in den Systemvariablen des Datenladeskripts festgelegte oder vom Betriebssystem vorgegebene Zahlenformat verwendet.

Beispiele und Ergebnisse:
In diesem Beispiel wird das Datumsformat **M/D/YYYY** verwendet. Das Datumsformat wird im Befehl **SET DateFormat** oben im Datenladeskript angegeben.

Fügen Sie Ihrer App dieses Beispielskript hinzu und führen Sie es aus.

```
Load *,

Num(Date#(StringDate)) as Date;

LOAD * INLINE [

StringDate

8/7/97

8/6/1997

]
```

Falls Sie eine Tabelle mit **StringDate**und **Date** als Dimensionen erstellen, sieht das Ergebnis folgendermaßen aus:

Results

StringDate	Datum
8/7/97	35649
8/6/1997	35648

Interval#

Interval#() evaluiert eine Textformel als Zeitintervall und verwendet dafür das standardmäßig vorgegebene Format des Betriebssystems oder das Format, das im zweiten Argument angegeben ist, wenn dies zur Verfügung steht.

Syntax:

```
Interval#(text[, format])
```

Rückgabe Datentyp: dual

Argumente:

Argumente

Argument	Beschreibung
text	Der zu evaluierende Textstring.
format	Ein String beschreibt das erwartete Eingabeformat für die Umwandlung eines Strings in ein numerisches Intervall. Wenn nicht angegeben, werden das kurze Datumsformat, das Zeitformat und das Dezimaltrennzeichen wie im Betriebssystem festgelegt verwendet.

Die Funktion **interval#** konvertiert ein Textzeitintervall in ein numerisches Äquivalent.

Beispiele und Ergebnisse:
Bei den nachstehenden Beispielen gehen wir von folgenden Einstellungen des Betriebssystems aus:

- Kurzes Datumsformat: YY-MM-DD
- Zeitformat: M/D/YY
- Dezimaltrennzeichen: .

Results

Beispiel	Ergebnis
`Interval#( A, 'D hh:mm' )` dabei gilt: A='1 09:00'	1.375

Money#

Money#() wandelt einen Textstring in einen Geldwert um und verwendet dafür, wenn kein Formatcode zur Verfügung steht, das vorgegebene Format des Ladeskripts oder des Betriebssystems. Angepasste Dezimal- und Tausendertrennzeichen sind optionale Parameter.

Syntax:

```
Money#(text[, format[, dec_sep [, thou_sep ] ] ])
```

Rückgabe Datentyp: dual

Argumente:

Argumente

Argument	Beschreibung
text	Der zu evaluierende Textstring.
format	Ein String beschreibt das erwartete Eingabeformat für die Umwandlung eines Strings in ein numerisches Intervall. Ist kein String angegeben, wird das vom Betriebssystem vorgegebene Währungsformat verwendet.
dec_sep	String zur Angabe des Dezimaltrennzeichens. Ist kein String angegeben, wird der vom Datenladeskript vorgegebene Wert "MoneyDecimalSep" verwendet.
thou_sep	String zur Angabe des Tausendertrennzeichens. Ist kein String angegeben, wird der vom Datenladeskript vorgegebene Wert "MoneyThousandSep" verwendet.

Die **money#**-Funktion verhält sich im Allgemeinen genauso wie die **num#**-Funktion, benutzt aber die vom System vorgegebenen oder durch die Skriptvariablen definierten Dezimal- und Tausendertrennzeichen für Geldbeträge.

Beispiele und Ergebnisse:
Bei den nachstehenden Beispielen gehen wir von folgenden Einstellungen des Betriebssystems aus:

- Standardeinstellung 1 für Währungsformat: kr # ##0,00
- Standardeinstellung 2 für Währungsformat: $ #,##0.00

```
Money#(A , '# ##0,00 kr' )
```
, wobei A=35 648,37 kr

Results

Results	Standardformat 1	Standardformat 2
String	35 648.37 €	35 648.37 €
Zahl	35648.37	3564837

```
Money#( A, ' $#', '.', ',' )
```
, wobei A= $35.648,37

Results

Results	Standardformat 1	Standardformat 2
String	$35,648.37	$35,648.37
Zahl	35648.37	35648.37

Num#

Num#() interpretiert einen Textstring als Zahlenwert, konvertiert also den Eingabestring in eine Zahl mit dem Format, das im zweiten Parameter angegeben ist. Wenn der zweite Parameter ausgelassen wird, werden die Dezimal- und Tausendertrennzeichen verwendet, die im Datenladeskript festgelegt sind. Angepasste Dezimal- und Tausendertrennzeichen sind optionale Parameter.

Syntax:

```
Num#(text[, format[, dec_sep [, thou_sep ] ] ])
```

Rückgabe Datentyp: dual

Die Funktion **Num#()** gibt einen dualen Wert mit dem String- und dem Zahlenwert zurück. Die Funktion nimmt die Textdarstellung der Eingabeformel und generiert eine Zahl. Das Format der Zahl wird nicht geändert. Die Ausgabe ist genau so formatiert wie die Eingabe.

Argumente:

Argumente

Argument	Beschreibung
text	Der zu evaluierende Textstring.
format	String zur Angabe des Zahlenformats im ersten Parameter. Ist kein String angegeben, werden die Dezimal- und Tausendertrennzeichen verwendet, die im Datenladeskript festgelegt sind.
dec_sep	String zur Angabe des Dezimaltrennzeichens. Ist kein String angegeben, wird der vom Datenladeskript vorgegebene Wert der Variablen DecimalSep verwendet.
thou_sep	String zur Angabe des Tausendertrennzeichens. Ist kein String angegeben, wird der vom Datenladeskript vorgegebene Wert der Variablen ThousandSep verwendet.

Beispiele und Ergebnisse:
Die folgende Tabelle zeigt das Ergebnis von *Num#(A, '#', '.' , ',')* für verschiedene Werte von A.

Ergebnisse

A	String-Darstellung	Numerischer Wert (hier mit Dezimalpunkt dargestellt)
35,648.31	35,648.31	35648.31
35 648.312	35 648.312	35648.312
35.648,3123	35.648,3123	-
35 648,31234	35 648,31234	-

Text

Text() interpretiert die Formel als Text, auch wenn eine numerische Interpretation möglich ist.

Syntax:

```
Text (expr)
```

Rückgabe Datentyp: dual

Beispiel:

```
Text( A )
```
wobei A=1,234

Results

String	Zahl
1234	-

Beispiel:

```
Text( pi( ) )
```

Results

String	Zahl
3.1415926535898	-

Time#

Time#() evaluiert eine Formel als Zeitwert und verwendet dafür, wenn kein Formatcode zur Verfügung steht, das vorgegebene Zeitformat des Datenladeskripts oder des Betriebssystems..

Syntax:

```
time#(text[, format])
```

Rückgabe Datentyp: dual

Argumente:

Argumente

Argument	Beschreibung
text	Der zu evaluierende Textstring.
format	String, der das Format des auszuwertenden Textstrings beschreibt. Wenn nicht angegeben, werden das kurze Datumsformat, das Zeitformat und das Dezimaltrennzeichen wie im Betriebssystem festgelegt verwendet.

Beispiel:

- Standardeinstellung 1 für Zeitformat: hh:mm:ss
- Standardeinstellung 2 für Zeitformat: hh.mm.ss

```
time#( A )
```
wobei A=09:00:00

Results

Results	Standardformat 1	Standardformat 2
String:	09:00:00	09:00:00
Zahl:	0.375	-

Beispiel:

- Standardeinstellung 1 für Zeitformat: hh:mm:ss
- Standardeinstellung 2 für Zeitformat: hh.mm.ss

```
time#( A, 'hh.mm' )
```
wobei A=09.00

Results

Results	Standardformat 1	Standardformat 2
String:	09.00	09.00
Zahl:	0.375	0.375

Timestamp#

Timestamp#() evaluiert eine Formel als Daten- und Zeitwert und verwendet dafür, wenn kein Formatcode zur Verfügung steht, das vorgegebene Zeitstempelformat des Datenladeskripts oder des Betriebssystems.

Syntax:

```
timestamp#(text[, format])
```

Rückgabe Datentyp: dual

Argumente:

Argumente

Argument	Beschreibung
text	Der zu evaluierende Textstring.

Argument	Beschreibung
format	String, der das Format des auszuwertenden Textstrings beschreibt. Wenn nicht angegeben, werden das kurze Datumsformat, das Zeitformat und das Dezimaltrennzeichen wie im Betriebssystem festgelegt verwendet. ISO 8601 wird für Zeitstempel unterstützt.

Beispiel:

In diesem Beispiel wird das Datumsformat **M/D/YYYY** verwendet. Das Datumsformat wird im Befehl **SET DateFormat** oben im Datenladeskript angegeben.

Fügen Sie Ihrer App dieses Beispielskript hinzu und führen Sie es aus.

```
Load *,
Timestamp(Timestamp#(String)) as TS;
LOAD * INLINE [
String
2015-09-15T12:13:14
1952-10-16T13:14:00+0200
1109-03-01T14:15
];
```

Falls Sie eine Tabelle mit **String** und **TS** als Dimensionen erstellen, sieht das Ergebnis folgendermaßen aus:

Results

String	TS
2015-09-15T12:13:14	9/15/2015 12:13:14 PM
1952-10-16T13:14:00+0200	10/16/1952 11:14:00 AM
1109-03-01T14:15	3/1/1109 2:15:00 PM

8.16 Inter-Record-Funktionen

Inter-Record-Funktionen werden in folgenden Fällen verwendet:

- Im Datenladeskript, wenn zur Evaluation des aktuellen Datensatzes Werte aus vorangehenden Datensätzen herangezogen werden sollen.
- In Diagrammformeln, wenn ein weiterer Wert des Datensatzes einer Visualisierung benötigt wird.

Das Sortieren nach y-Werten in Diagrammen oder nach Formelspalten in Tabellen ist nicht zulässig, wenn eine datensatzübergreifende Diagrammfunktion in einer der Diagrammformeln verwendet wird. Diese Sortierungsoptionen werden daher automatisch deaktiviert. Wenn Sie eine datensatzübergreifende Diagrammfunktion in einer Visualisierung oder Tabelle verwenden, wird die Sortierung der Visualisierung auf die sortierte Eingabe der datensatzübergreifenden Funktion zurückgesetzt. Diese Einschränkung gilt nicht für die entsprechende Skriptfunktion, falls vorhanden.

Definitionen für auf sich selbst verweisende Formeln können nur zuverlässig in Tabellen mit weniger als 100 Zeilen erstellt werden, aber dies kann sich je nach der Hardware unterscheiden, auf der die Qlik Engine ausgeführt wird.

Zeilenfunktionen

Diese Funktionen können ausschließlich in den Formeln von Diagrammen verwendet werden.

Above

Above() interpretiert eine Formel in einer Zeile über der aktuellen Zeile innerhalb eines Spaltenabschnitts in einer Tabelle. Die Zeile, für welche die Berechnung erfolgt, hängt vom Wert von **offset** ab; ist dieser vorhanden, wird standardmäßig die Zeile direkt darüber verwendet. In Diagrammen wird anders als in Tabellen mit **Above()** die Zeile über der aktuellen Zeile im entsprechenden Tabellendiagramm interpretiert.

```
Above - Diagrammfunktion([TOTAL [<fld{,fld}>]] expr [ , offset [,count]])
```

Below

Below() interpretiert eine Formel in einer Zeile unter der aktuellen Zeile innerhalb eines Spaltenabschnitts in einer Tabelle. Die Zeile, für welche die Berechnung erfolgt, hängt vom Wert von **offset** ab; ist dieser vorhanden, wird standardmäßig die Zeile direkt darunter verwendet. In Diagrammen wird anders als in Tabellen mit **Below()** die Zeile unter der aktuellen Spalte im entsprechenden Tabellendiagramm interpretiert.

```
Below - Diagrammfunktion([TOTAL[<fld{,fld}>]] expression [ , offset [,count
]])
```

Bottom

Bottom() interpretiert eine Formel in der letzten (untersten) Zeile eines Spaltenabschnitts in einer Tabelle. Die Zeile, für welche die Berechnung erfolgt, hängt vom Wert von **offset** ab; ist dieser vorhanden, wird standardmäßig die unterste Zeile verwendet. In Diagrammen erfolgt anders als in Tabellen die Berechnung für die letzte Zeile der aktuellen Spalte im entsprechenden Tabellendiagramm.

```
Bottom - Diagrammfunktion([TOTAL[<fld{,fld}>]] expr [ , offset [,count ]])
```

Top

Top() interpretiert eine Formel in der ersten (obersten) Zeile eines Spaltenabschnitts in einer Tabelle. Die Zeile, für welche die Berechnung erfolgt, hängt vom Wert von **offset** ab; ist dieser vorhanden, wird standardmäßig die oberste Zeile verwendet. In Diagrammen erfolgt anders als in Tabellen mit **Top()** die Berechnung für die erste Zeile der aktuellen Spalte im entsprechenden Tabellendiagramm.

```
Top - Diagrammfunktion([TOTAL [<fld{,fld}>]] expr [ , offset [,count ]])
```

NoOfRows

NoOfRows() liefert die Anzahl der Zeilen im aktuellen Spaltenabschnitt in einer Tabelle. In Bitmap-Diagrammen liefert **NoOfRows()** die Zahl der Zeilen im entsprechenden Äquivalent zum Tabellendiagramm.

```
NoOfRows - Diagrammfunktion([TOTAL])
```

Spaltenfunktionen

Diese Funktionen können ausschließlich in den Formeln von Diagrammen verwendet werden.

Column

Column() liefert den Wert aus der Spalte **ColumnNo** in einem Tabellendiagramm, ohne die Dimensionen zu berücksichtigen. So liefert **Column(2)** den Wert der zweiten Kennzahlspalte.

```
Column - Diagrammfunktion(ColumnNo)
```

Dimensionality

Dimensionality() liefert für die aktuelle Zeile die Anzahl der Dimensionen. Im Fall von Pivottabellen liefert die Funktion die Gesamtzahl der Dimensionsspalten ohne aggregierten Inhalt, d. h. ohne Partialsummen und ausgeblendete Dimensionen.

```
Dimensionality - Diagrammfunktion ( )
```

Secondarydimensionality

SecondaryDimensionality() liefert die Zahl der Dimensionszeilen der Pivottabelle ohne aggregierten Inhalt, das heißt ohne Partialsummen und ausgeblendete Dimensionen. Diese Funktion entspricht der Funktion **dimensionality()** in horizontalen Pivottabellen.

```
SecondaryDimensionality - Diagrammfunktion ( )
```

Feldfunktionen

FieldIndex

FieldIndex() liefert die Position des Wertes **value** im Feld **field_name** (nach Lade-Reihenfolge).

```
FieldIndex (field_name , value)
```

FieldValue

FieldValue() liefert den Wert an der (nach Lade-Reihenfolge) **elem_no**-ten Position des Feldes **field_name**.

```
FieldValue (field_name , elem_no)
```

FieldValueCount

FieldValueCount() ist eine **integer**-Funktion, die die Anzahl der distinkten Werte in einem Feld zurückgibt.

```
FieldValueCount (field_name)
```

Pivottabellenfunktionen

Diese Funktionen können ausschließlich in den Formeln von Diagrammen verwendet werden.

After

After() liefert das Ergebnis einer Formel, berechnet anhand der Dimensionswerte der nachfolgenden Formelspalte innerhalb desselben Zeilensegments der Pivottabelle.

```
After - Diagrammfunktion([TOTAL] expression [ , offset [,n]])
```

Before

Before() liefert das Ergebnis einer Formel, berechnet anhand der Dimensionswerte der vorhergehenden Formelspalte innerhalb desselben Zeilenabschnitts der Pivottabelle.

```
Before - Diagrammfunktion([TOTAL] expression [ , offset [,n]])
```

First

First() liefert das Ergebnis einer Formel, berechnet anhand der Dimensionswerte der ersten Formelspalte des Zeilenabschnitts der Pivottabelle. Diese Funktion ist ausschließlich für Pivottabellen vorgesehen und liefert in allen anderen Diagrammtypen NULL.

```
First - Diagrammfunktion([TOTAL] expression [ , offset [,n]])
```

Last

Last() liefert das Ergebnis einer Formel, berechnet anhand der Dimensionswerte der letzten Formelspalte des Zeilenabschnitts der Pivottabelle. Diese Funktion ist ausschließlich für Pivottabellen vorgesehen und liefert in allen anderen Diagrammtypen NULL.

```
Last - Diagrammfunktion([TOTAL] expression [ , offset [,n]])
```

ColumnNo

ColumnNo() liefert die Nummer der aktuellen Spalte innerhalb des Zeilenabschnitts der Pivottabelle. Die erste Spalte trägt die Nummer 1.

```
ColumnNo - Diagrammfunktion([TOTAL])
```

NoOfColumns

NoOfColumns() liefert die Anzahl der Spalten innerhalb des Zeilensegments in einer Pivottabelle.

```
NoOfColumns - Diagrammfunktion([TOTAL])
```

Inter-Record-Funktionen im Datenladeskript

Exists

Exists() bestimmt, ob ein spezifischer Feldwert bereits in das Feld im Datenladeskript geladen wurde. Die Funktion gibt TRUE oder FALSE zurück, und kann deshalb in der **where**-Bedingung eines **LOAD**-Befehls oder eines **IF**-Befehls verwendet werden.

```
Exists (field_name [, expr])
```

LookUp

Lookup() sucht in einer bereits geladenen Tabelle und liefert den Wert des Feldes **field_name**, der dem ersten Auftreten des Werts **match_field_value** im Feld **match_field_name** zugehörig ist. Bei der Tabelle kann es sich um die aktuelle Tabelle oder eine andere zuvor geladene Tabelle handeln.

```
LookUp (field_name, match_field_name, match_field_value [, table_name])
```

Peek

Peek() gibt den Wert eines Feldes in einer Tabelle für eine Zeile zurück, die bereits geladen wurde. Die Zeilennummer kann wie die Tabelle festgelegt werden. Wenn keine Zeilennummer angegeben ist, wird der letzte zuvor geladene Datensatz verwendet.

```
Peek (field_name[, row_no[, table_name ] ])
```

Previous

Previous() liefert den Wert der **expr**-Formel, wobei für die Berechnung Daten aus dem letzten Datensatz verwendet werden, der nicht durch einen **where**-Zusatz ausgeschlossen wurde. Im ersten Datensatz einer internen Tabelle liefert diese Funktion NULL.

```
Previous (page 1340)(expr)
```

Siehe auch:

- *Bereichsfunktionen (page 1361)*

Above - Diagrammfunktion

Above() interpretiert eine Formel in einer Zeile über der aktuellen Zeile innerhalb eines Spaltenabschnitts in einer Tabelle. Die Zeile, für welche die Berechnung erfolgt, hängt vom Wert von **offset** ab; ist dieser vorhanden, wird standardmäßig die Zeile direkt darüber verwendet. In Diagrammen wird anders als in Tabellen mit **Above()** die Zeile über der aktuellen Zeile im entsprechenden Tabellendiagramm interpretiert.

Syntax:

```
Above([TOTAL] expr [ , offset [,count]])
```

Rückgabe Datentyp: dual

Argumente:

Argumente

Argument	Beschreibung
expr	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.
offset	Ist ein offsetn größer als 0 angegeben, wird die Formel anhand der Werte von n Zeilen weiter oben berechnet. Bei einem Startwert von "0" wird die Formel anhand der Werte der aktuellen Zeile berechnet. Bei einem negativen Startwert verhält sich die Funktion Above wie die Funktion Below mit dem entsprechenden positiven Startwert.
count	Ist ein drittes Argument **count** größer als 1 angegeben, liefert die Funktion eine Menge von **count**-Werten, berechnet anhand der **count**-Zeilen oberhalb bis einschließlich der aktuellen Zeile. In diesem Formular kann die Funktion als Argument für eine der speziellen Abschnittsfunktionen dienen. *Bereichsfunktionen (page 1361)*
TOTAL	Wenn das Diagramm nur eine Dimension hat oder die Formel mit dem Zusatz **TOTAL** als Argument versehen ist, entspricht der Spaltenabschnitt der gesamten Spalte.

In der ersten Zeile eines Spaltenabschnitts ist das Ergebnis NULL, da davor keine Zeile existiert.

Ein Spaltenabschnitt ist als aufeinanderfolgende Teilmenge an Zellen definiert, welche dieselben Werte für die Dimensionen in der aktuellen Sortierreihenfolge besitzen. Inter-Record-Diagrammfunktionen werden im Spaltenabschnitt berechnet, wobei die rechts außen stehende Dimension im entsprechenden Tabellendiagramm ausgeschlossen wird. Befindet sich nur eine Dimension im Diagramm oder wird der Zusatz TOTAL angegeben, erfolgt die Berechnung über die gesamte Tabelle.

Hat das Diagramm dagegen mehrere vertikale Dimensionen, so umfasst der Spaltenabschnitt nur Zeilen, deren Werte in allen Dimensionen mit Ausnahme der in der Priorität der Sortierfolgen letzten Dimension übereinstimmen.

Beschränkungen:

- Die Rekursion liefert NULL.
- Das Sortieren nach y-Werten in Diagrammen oder nach Formelspalten in Tabellen ist nicht zulässig, wenn diese Diagrammfunktion in einer der Diagrammformeln verwendet wird. Diese Sortierungsoptionen werden daher automatisch deaktiviert. Wenn Sie diese Diagrammfunktion in einer Visualisierung oder Tabelle verwenden, wird die Sortierung der Visualisierung auf die sortierte Eingabe dieser Funktion zurückgesetzt.

Beispiele und Ergebnisse:

Example 1:

Tabellenvisualisierung für Beispiel 1

Customer	Sum([Sales])	Above(Sum(Sales))	Sum(Sales)+Above(Sum(Sales))	Above offset 3	Higher?
	2566	-	-	-	-
Astrida	587	-	-	-	-
Betacab	539	587	1126	-	-
Canutility	683	539	1222	-	Higher
Divadip	757	683	1440	1344	Higher

Im Screenshot der in diesem Beispiel gezeigten Tabelle wird die Tabellenvisualisierung aus der Dimension **Customer** und den folgenden Kennzahlen erstellt: `Sum(Sales)` und `Above(Sum(Sales))`.

Die Spalte Above(Sum(Sales)) gibt NULL für die Zeile **Customer** mit **Astrida** aus, da dies die erste Zeile ist. Das Ergebnis für die Zeile **Betacab** zeigt den Wert von Sum(Sales) für **Astrida** an, das Ergebnis für **Canutility** zeigt den Wert von **Sum(Sales)** für **Betacab** an usw.

Für die Spalte mit der Beschriftung Sum(Sales)+Above(Sum(Sales)) wird in der Zeile für **Betacab** das Ergebnis der Addition der Werte **Sum(Sales)** für die Zeilen **Betacab** + **Astrida** (539+587) angezeigt. Das Ergebnis für die Zeile**Canutility** ist die Addition der Werte **Sum(Sales)** für **Canutility** + **Betacab** (683+539).

Die Kennzahl mit der Beschriftung Above offset 3, die mithilfe der Formel `Sum(Sales)+Above(Sum(Sales), 3)` erstellt wurde, verfügt über das Argument **offset** mit der Einstellung 3 und wählt den Wert drei Zeilen oberhalb der aktuellen Zeile aus. Sie fügt den Wert **Sum(Sales)** für die aktuelle Auswahl von **Customer** zum Wert für **Customer** drei Zeilen darüber hinzu. Als Werte für die ersten drei Zeilen **Customer** werden NULL-Werte zurückgegeben.

In der Tabelle werden auch komplexere Kennzahlen angezeigt: eine erstellt aus `Sum(Sales)+Above(Sum(Sales))` und eine beschriftet mit **Higher?**, die aus `IF(Sum(Sales)>Above(Sum(Sales)), 'Higher')` erstellt wird.

Diese Funktion kann auch in anderen Diagrammen als Tabellen wie zum Beispiel in Balkendiagrammen verwendet werden.

Bei anderen Diagrammtypen wandeln Sie das Diagramm in ein entsprechendes Tabellendiagramm um, damit Sie einfach nachvollziehen können, auf welche Zeile sich die Funktion bezieht.

Example 2:

In den Screenshots der in diesem Beispiel gezeigten Tabellen wurden den Visualisierungen weitere Dimensionen hinzugefügt: **Month** und **Product**. Bei Diagrammen mit mehr als einer Dimension hängt das Ergebnis der Formeln, welche die Funktionen **Above**, **Below**, **Top** und **Bottom** beinhalten, von der Reihenfolge ab, in der die Spaltendimensionen von Qlik Sense sortiert werden. Qlik Sense evaluiert die Funktionen auf Grundlage der Spaltenabschnitte, die sich aus der zuletzt sortierten Dimension ergeben haben. Die Sortierreihenfolge der Spalten kann im Eigenschaftsfenster unter **Sortierung** festgelegt werden und stimmt nicht zwangsläufig mit der Reihenfolge überein, in der die Spalten in der Tabelle angezeigt werden.

Im folgenden Screenshot der Tabellenvisualisierung aus Beispiel 2 ist die letzte sortierte Dimension **Month**, sodass die Funktion **Above** die Berechnung auf der Grundlage von Monaten durchführt. Für jeden Wert **Product** gibt es pro Monat (**Jan** bis **Aug**) eine Ergebnisserie – einen Spaltenabschnitt. Darauf folgt eine Ergebnisserie für den nächsten Spaltenabschnitt: für jeden **Month** für das nächste **Product**. Für jeden Wert **Customer** wird jeweils pro **Product** ein eigener Spaltenabschnitt ausgegeben.

Tabellenvisualisierung für Beispiel 2

Customer	Product	Month	Sum([Sales])	Above(Sum(Sales))
			2566	-
Astrida	AA	Jan	46	-
Astrida	AA	Feb	60	46
Astrida	AA	Mar	70	60
Astrida	AA	Apr	13	70
Astrida	AA	May	78	13
Astrida	AA	Jun	20	78
Astrida	AA	Jul	45	20
Astrida	AA	Aug	65	45

Example 3:

Im Screenshot der Tabellenvisualisierung für Beispiel 3 ist die letzte sortierte Dimension **Product**. Dies erfolgt durch Verschieben der Dimension Product an Position 3 in der Dialogseite für die Sortierung im Eigenschaftsfenster. Die Funktion **Above** wird für jedes **Product** berechnet. Da nur zwei Produkte vorhanden sind, **AA** und **BB**, kann in beiden Ergebnisserien nur ein Ergebnis nicht NULL sein. In Zeile **BB** für den Monat **Jan** ist der Wert für **Above(Sum(Sales))** 46. Für Zeile **AA** ist der Wert Null. Der Wert jeder Zeile **AA** für einen Monat ist immer Null, da kein Wert von **Product** über AA vorhanden ist. Die zweite Ergebnisserie wird für **AA** und **BB** für den Monat **Feb** auf Grundlage des Werts **Customer**, **Astrida**, berechnet. Wurden alle Monate für **Astrida** berechnet, wird die Sequenz für den zweiten **Customer**Betacab wiederholt, dann ggf. für den dritten usw.

Tabellenvisualisierung für Beispiel 3

Customer	Product	Month	Sum([Sales])	Above(Sum(Sales))
			2566	-
Astrida	AA	Jan	46	-
Astrida	BB	Jan	46	46
Astrida	AA	Feb	60	-
Astrida	BB	Feb	60	60
Astrida	AA	Mar	70	-
Astrida	BB	Mar	70	70
Astrida	AA	Apr	13	-
Astrida	BB	Apr	13	13

Beispiel 4

Example 4:	**Ergebnis**
Die Funktion Above kann als Input für die Abschnittsfunktionen verwendet werden. Hier ein Beispiel: `RangeAvg (Above(Sum (Sales),1,3))`.	In den Argumenten für die Funktion Above() wird offset auf 1 und count auf 3 gesetzt. Die Funktion ermittelt die Ergebnisse der Formel Sum(Sales) zu den drei Zeilen unmittelbar über der aktuellen Zeile im Spaltenabschnitt (sofern Zeilen vorhanden sind). Diese drei Werte werden als Input für die Funktion RangeAvg() verwendet, die den Durchschnitt der Werte in der gelieferten Menge an Zahlen ermittelt. Eine Tabelle mit Customer als Dimension liefert folgende Ergebnisse für die Formel RangeAvg(). Astrida - Betacab 587 Canutility 563 Divadip: 603

In Beispielen verwendete Daten:

```
Monthnames:
LOAD *, Dual(MonthText,MonthNumber) as Month INLINE [
MonthText, MonthNumber
Jan, 1
Feb, 2
Mar, 3
Apr, 4
May, 5
Jun, 6
Jul, 7
Aug, 8
Sep, 9
Oct, 10
Nov, 11
Dec, 12
];

Sales2013:
Crosstable (MonthText, Sales) LOAD * inline [
Customer|Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec
Astrida|46|60|70|13|78|20|45|65|78|12|78|22
Betacab|65|56|22|79|12|56|45|24|32|78|55|15
Canutility|77|68|34|91|24|68|57|36|44|90|67|27
Divadip|57|36|44|90|67|27|57|68|47|90|80|94
] (delimiter is '|');
```

Siehe auch:

- *Below - Diagrammfunktion (page 1308)*
- *Bottom - Diagrammfunktion (page 1311)*
- *Top - Diagrammfunktion (page 1341)*
- *RangeAvg (page 1364)*

Below - Diagrammfunktion

Below() interpretiert eine Formel in einer Zeile unter der aktuellen Zeile innerhalb eines Spaltenabschnitts in einer Tabelle. Die Zeile, für welche die Berechnung erfolgt, hängt vom Wert von **offset** ab; ist dieser vorhanden, wird standardmäßig die Zeile direkt darunter verwendet. In Diagrammen wird anders als in Tabellen mit **Below()** die Zeile unter der aktuellen Spalte im entsprechenden Tabellendiagramm interpretiert.

Syntax:

```
Below([TOTAL] expr [ , offset [,count ]])
```

Rückgabe Datentyp: dual

Argumente:

Argumente

Argument	Beschreibung
expr	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.
offset	Ist ein **offset**n größer als 1 angegeben, wird die Formel anhand der Werte von n Zeilen weiter unten ausgewertet. Bei einem Startwert von "0" wird die Formel anhand der Werte der aktuellen Zeile berechnet. Bei einem negativen Startwert verhält sich die Funktion **Below** wie die Funktion **Above** mit dem entsprechenden positiven Startwert.
count	Ist ein dritter Parameter **count** größer als 1 angegeben, liefert die Funktion eine Menge von **count**-Werten, berechnet anhand der **count**-Zeilen unterhalb bis einschließlich der aktuellen Zeile. In diesem Formular kann die Funktion als Argument für eine der speziellen Abschnittsfunktionen dienen. *Bereichsfunktionen (page 1361)*
TOTAL	Wenn das Diagramm nur eine Dimension hat oder die Formel mit dem Zusatz **TOTAL** als Argument versehen ist, entspricht der Spaltenabschnitt der gesamten Spalte.

In der letzten Zeile eines Spaltenabschnitts ist das Ergebnis NULL, da keine darauffolgende Zeile existiert.

Ein Spaltenabschnitt ist als aufeinanderfolgende Teilmenge an Zellen definiert, welche dieselben Werte für die Dimensionen in der aktuellen Sortierreihenfolge besitzen. Inter-Record-Diagrammfunktionen werden im Spaltenabschnitt berechnet, wobei die rechts außen stehende Dimension im entsprechenden Tabellendiagramm ausgeschlossen wird. Befindet sich nur eine Dimension im Diagramm oder wird der Zusatz TOTAL angegeben, erfolgt die Berechnung über die gesamte Tabelle.

Hat das Diagramm dagegen mehrere vertikale Dimensionen, so umfasst der Spaltenabschnitt nur Zeilen, deren Werte in allen Dimensionen mit Ausnahme der in der Priorität der Sortierfolgen letzten Dimension übereinstimmen.

Beschränkungen:

- Die Rekursion liefert NULL.
- Das Sortieren nach y-Werten in Diagrammen oder nach Formelspalten in Tabellen ist nicht zulässig, wenn diese Diagrammfunktion in einer der Diagrammformeln verwendet wird. Diese Sortierungsoptionen werden daher automatisch deaktiviert. Wenn Sie diese Diagrammfunktion in einer Visualisierung oder Tabelle verwenden, wird die Sortierung der Visualisierung auf die sortierte Eingabe dieser Funktion zurückgesetzt.

Beispiele und Ergebnisse:

Example 1:

Tabellenvisualisierung für Beispiel 1

Customer	Sum([Sales])	Below(Sum(Sales))	Sum(Sales)+Below(Sum(Sales))	Below + Offset 3	Higher
	2566	-	-	-	-
Astrida	587	539	1126	1344	Higher
Betacab	539	683	1222	-	-
Canutility	683	757	1440	-	-
Divadip	757	-	-	-	-

In der im Screenshot für Beispiel 1 gezeigten Tabelle wird die Tabellenvisualisierung aus der Dimension **Customer** und den folgenden Kennzahlen erstellt: `Sum(Sales)` und `Below(Sum(Sales))`.

Die Spalte **Below(Sum(Sales))** gibt NULL für die Zeile **Customer** mit **Divadip** aus, da dies die letzte Zeile ist. Das Ergebnis für die Zeile **Canutility** zeigt den Wert von Sum(Sales) für **Divadip** an, das Ergebnis für **Betacab** zeigt den Wert von **Sum(Sales)** für **Canutility** an usw.

In der Tabelle werden auch komplexere Kennzahlen angezeigt, was in folgenden Spalten ersichtlich ist: `Sum(Sales)+Below(Sum(Sales))`, **Below +Offset 3** und **Higher?**. Diese Formeln funktionieren in den folgenden Absätzen wie beschrieben.

Für die Spalte mit der Beschriftung **Sum(Sales)+Below(Sum(Sales))** wird in der Zeile für **Astrida** das Ergebnis der Addition der Werte **Sum(Sales)** für die Zeilen **Betacab** + **Astrida** (539+587) angezeigt. Das Ergebnis für die Zeile **Betacab** ist die Addition der Werte **Sum(Sales)** für **Canutility** + **Betacab** (539+683).

Die Kennzahl mit der Beschriftung **Below +Offset 3**, die mithilfe der Formel `Sum(Sales)+Below(Sum(Sales), 3)` erstellt wurde, verfügt über das Argument **offset** mit der Einstellung 3 und wählt den Wert drei Zeilen unter der aktuellen Zeile aus. Sie fügt den Wert **Sum(Sales)** für die aktuelle Auswahl von **Customer** zum Wert des Felds **Customer** drei Zeilen darunter hinzu. Die Werte der untersten drei Zeilen **Customer** sind NULL-Werte.

Die Kennzahl mit der Bezeichnung **Higher?** wird aus dieser Formel erstellt:`IF(Sum(Sales)>Below(Sum(Sales)), 'Higher')`. Dadurch werden die Werte der aktuellen Zeile in der Kennzahl **Sum(Sales)** mit der Zeile darunter verglichen. Wenn die aktuelle Zeile ein höherer Wert ist, erfolgt die Ausgabe des Texts "Higher".

Diese Funktion kann auch in anderen Diagrammen als Tabellen wie zum Beispiel in Balkendiagrammen verwendet werden.

Bei anderen Diagrammtypen wandeln Sie das Diagramm in ein entsprechendes Tabellendiagramm um, damit Sie einfach nachvollziehen können, auf welche Zeile sich die Funktion bezieht.

Bei Diagrammen mit mehr als einer Dimension hängt das Ergebnis der Formeln, welche die Funktionen **Above**, **Below**, **Top** und **Bottom** beinhalten, von der Reihenfolge ab, in der die Spaltendimensionen von Qlik Sense sortiert werden. Qlik Sense evaluiert die Funktionen auf Grundlage der Spaltenabschnitte, die sich aus der zuletzt sortierten Dimension ergeben haben. Die Sortierreihenfolge der Spalten kann im Eigenschaftsfenster unter **Sortierung** festgelegt werden und stimmt nicht zwangsläufig mit der Reihenfolge überein, in der die Spalten in der Tabelle angezeigt werden.Siehe Beispiel: 2 in der Funktion **Above** für weitere Details.

Beispiel 2

Example 2:	**Ergebnis**
Die Funktion **Below** kann als Input für die Abschnittsfunktionen verwendet werden. Hier ein Beispiel: `RangeAvg (Below(Sum(Sales),1,3))`.	In den Argumenten für die Funktion **Below()** wird offset auf 1 und count auf 3 gesetzt. Die Funktion ermittelt die Ergebnisse der Formel **Sum(Sales)** zu den drei Zeilen unmittelbar unter der aktuellen Zeile im Spaltenabschnitt (sofern Zeilen vorhanden sind). Diese drei Werte werden als Input für die Funktion RangeAvg() verwendet, die den Durchschnitt der Werte in der gelieferten Menge an Zahlen ermittelt. Eine Tabelle mit **Customer** als Dimension liefert folgende Ergebnisse für die Formel RangeAvg().
	Astrida 659.67 Betacab 720 Canutility 757 Divadip: -

In Beispielen verwendete Daten:

```
Monthnames:
LOAD *, Dual(MonthText,MonthNumber) as Month INLINE [
MonthText, MonthNumber
Jan, 1
Feb, 2
Mar, 3
Apr, 4
May, 5
Jun, 6
Jul, 7
Aug, 8
Sep, 9
Oct, 10
Nov, 11
Dec, 12
];

Sales2013:
Crosstable (MonthText, Sales) LOAD * inline [
Customer|Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec
Astrida|46|60|70|13|78|20|45|65|78|12|78|22
Betacab|65|56|22|79|12|56|45|24|32|78|55|15
Canutility|77|68|34|91|24|68|57|36|44|90|67|27
Divadip|57|36|44|90|67|27|57|68|47|90|80|94
] (delimiter is '|');
```

Siehe auch:

- *Above - Diagrammfunktion (page 1303)*
- *Bottom - Diagrammfunktion (page 1311)*
- *Top - Diagrammfunktion (page 1341)*
- *RangeAvg (page 1364)*

Bottom - Diagrammfunktion

Bottom() interpretiert eine Formel in der letzten (untersten) Zeile eines Spaltenabschnitts in einer Tabelle. Die Zeile, für welche die Berechnung erfolgt, hängt vom Wert von **offset** ab; ist dieser vorhanden, wird standardmäßig die unterste Zeile verwendet. In Diagrammen erfolgt anders als in Tabellen die Berechnung für die letzte Zeile der aktuellen Spalte im entsprechenden Tabellendiagramm.

Syntax:

```
Bottom([TOTAL] expr [ , offset [,count ]])
```

Rückgabe Datentyp: dual

Argumente:

Argumente

Argument	Beschreibung
expr	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.
offset	Ist ein **offset**n größer als 1 angegeben, wird die Formel anhand der Werte von n Zeilen berechnet, die über der letzten Zeile liegen. Bei einem negativen Startwert verhält sich die Funktion **Bottom** wie die Funktion **Top** mit dem entsprechenden positiven Startwert.
count	Ist ein dritter Parameter **count** größer als 1 angegeben, liefert die Funktion nicht einen einzelnen Wert, sondern eine Menge von **count**-Werten, berechnet anhand der letzten **count**-Zeilen des Spaltenabschnitts. In diesem Formular kann die Funktion als Argument für eine der speziellen Abschnittsfunktionen dienen. *Bereichsfunktionen (page 1361)*
TOTAL	Wenn das Diagramm nur eine Dimension hat oder die Formel mit dem Zusatz **TOTAL** als Argument versehen ist, entspricht der Spaltenabschnitt der gesamten Spalte.

Ein Spaltenabschnitt ist als aufeinanderfolgende Teilmenge an Zellen definiert, welche dieselben Werte für die Dimensionen in der aktuellen Sortierreihenfolge besitzen. Inter-Record-Diagrammfunktionen werden im Spaltenabschnitt berechnet, wobei die rechts außen stehende Dimension im entsprechenden Tabellendiagramm ausgeschlossen wird. Befindet sich nur eine Dimension im Diagramm oder wird der Zusatz TOTAL angegeben, erfolgt die Berechnung über die gesamte Tabelle.

Hat das Diagramm dagegen mehrere vertikale Dimensionen, so umfasst der Spaltenabschnitt nur Zeilen, deren Werte in allen Dimensionen mit Ausnahme der in der Priorität der Sortierfolgen letzten Dimension übereinstimmen.

Beschränkungen:

- Die Rekursion liefert NULL.
- Das Sortieren nach y-Werten in Diagrammen oder nach Formelspalten in Tabellen ist nicht zulässig, wenn diese Diagrammfunktion in einer der Diagrammformeln verwendet wird. Diese Sortierungsoptionen werden daher automatisch deaktiviert. Wenn Sie diese Diagrammfunktion in einer Visualisierung oder Tabelle verwenden, wird die Sortierung der Visualisierung auf die sortierte Eingabe dieser Funktion zurückgesetzt.

Beispiele und Ergebnisse:

Tabellenvisualisierung für Beispiel 1

Customer	Sum([Sales])	Bottom(Sum(Sales))	Sum(Sales)+Bottom(Sum(Sales))	Bottom offset 3
	2566	**757**	**3323**	**3105**
Astrida	587	757	1344	1126
Betacab	539	757	1296	1078
Canutility	683	757	1440	1222
Divadip	757	757	1514	1296

Im Screenshot der in diesem Beispiel gezeigten Tabelle wird die Tabellenvisualisierung aus der Dimension **Customer** und den folgenden Kennzahlen erstellt: `Sum(Sales)` und `Bottom(Sum(Sales))`.

Die Spalte **Bottom(Sum(Sales))** gibt 757 für alle Zeilen zurück, weil dies der Wert für die untere Zeile ist: **Divadip**.

In der Tabelle werden auch komplexere Kennzahlen angezeigt: eine aus `Sum(Sales)+Bottom(Sum(Sales))` erstellte und eine mit **Bottom offset 3** beschriftete Kennzahl, die mithilfe der Formel `Sum(Sales)+Bottom(Sum(Sales), 3)` erstellt wird und über das Argument **offset** mit der Einstellung 3 verfügt. Sie fügt den Wert **Sum (Sales)** für die aktuelle Zeile zum Wert aus der dritten Zeile von der untersten Zeile hinzu, d. h. die aktuelle Zeile plus den Wert für **Betacab**.

Beispiel: 2

In den Screenshots der in diesem Beispiel gezeigten Tabellen wurden den Visualisierungen weitere Dimensionen hinzugefügt: **Month** und **Product**. Bei Diagrammen mit mehr als einer Dimension hängt das Ergebnis der Formeln, welche die Funktionen **Above**, **Below**, **Top** und **Bottom** beinhalten, von der Reihenfolge ab, in der die Spaltendimensionen von Qlik Sense sortiert werden. Qlik Sense evaluiert die Funktionen auf Grundlage der Spaltenabschnitte, die sich aus der zuletzt sortierten Dimension ergeben haben. Die Sortierreihenfolge der Spalten kann im Eigenschaftsfenster unter **Sortierung** festgelegt werden und stimmt nicht zwangsläufig mit der Reihenfolge überein, in der die Spalten in der Tabelle angezeigt werden.

In der ersten Tabelle wird die Formel auf Grundlage von **Month** und in der zweiten Tabelle auf Grundlage von **Product** berechnet. Die Kennzahl **End value** enthält die Formel `Bottom(Sum(Sales))`. Die unterste Zeile für **Month** ist Dec und der Wert für Dec beide Werte von im Screenshot dargestellten **Product** ist 22. (Um Platz zu sparen, wurden einige Zeilen aus dem Screenshot entfernt.)

Erste Tabelle für Beispiel 2. Der Wert von Bottom für die Kennzahl End value basiert auf Month (Dec).

Customer	Product	Month	Sum(Sales)	End value
			2566	-
Astrida	AA	Jan	46	22
Astrida	AA	Feb	60	22
Astrida	AA	Mar	70	22
Astrida	AA	Sep	78	22
Astrida	AA	Oct	12	22
Astrida	AA	Nov	78	22
Astrida	AA	Dec	22	22
Astrida	BB	Jan	46	22

Zweite Tabelle für Beispiel 2. Der Wert von Bottom für die Kennzahl End value basiert auf Product (BB für Astrida).

Customer	Product	Month	Sum(Sales)	End value
			2566	-
Astrida	AA	Jan	46	46
Astrida	BB	Jan	46	46
Astrida	AA	Feb	60	60
Astrida	BB	Feb	60	60
Astrida	AA	Mar	70	70
Astrida	BB	Mar	70	70
Astrida	AA	Apr	13	13
Astrida	BB	Apr	13	13

Siehe Beispiel: 2 in der Funktion **Above** für weitere Details.

Beispiel 3

Beispiel: 3	**Ergebnis**
Die Funktion **Bottom** kann als Input für die Abschnittsfunktionen verwendet werden. Hier ein Beispiel: `RangeAvg (Bottom(Sum(Sales),1,3))`.	In den Argumenten für die Funktion **Bottom()** wird offset auf 1 und count auf 3 gesetzt. Die Funktion ermittelt die Ergebnisse der Formel **Sum(Sales)** zu den drei Zeilen, beginnend bei der Zeile über der letzten Zeile im Spaltenabschnitt (weil offset=1), und den beiden Zeilen darüber (sofern Zeilen vorhanden sind). Diese drei Werte werden als Input für die Funktion RangeAvg() verwendet, die den Durchschnitt der Werte in der gelieferten Menge an Zahlen ermittelt. Eine Tabelle mit **Customer** als Dimension liefert folgende Ergebnisse für die Formel RangeAvg().
	Astrida 659.67 Betacab 659.67 Canutility 659.67 Divadip: 659.67

```
Monthnames:
LOAD *, Dual(MonthText,MonthNumber) as Month INLINE [
MonthText, MonthNumber
Jan, 1
Feb, 2
Mar, 3
Apr, 4
May, 5
Jun, 6
Jul, 7
Aug, 8
Sep, 9
Oct, 10
Nov, 11
Dec, 12
];

Sales2013:
Crosstable (MonthText, Sales) LOAD * inline [
Customer|Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec
Astrida|46|60|70|13|78|20|45|65|78|12|78|22
Betacab|65|56|22|79|12|56|45|24|32|78|55|15
Canutility|77|68|34|91|24|68|57|36|44|90|67|27
Divadip|57|36|44|90|67|27|57|68|47|90|80|94
] (delimiter is '|');
```

Siehe auch:

Top - Diagrammfunktion (page 1341)

Column - Diagrammfunktion

Column() liefert den Wert aus der Spalte **ColumnNo** in einem Tabellendiagramm, ohne die Dimensionen zu berücksichtigen. So liefert **Column(2)** den Wert der zweiten Kennzahlspalte.

Syntax:

```
Column(ColumnNo)
```

Rückgabe Datentyp: dual

Argumente:

Argumente

Argument	Beschreibung
ColumnNo	Die Nummer der Spalte mit einer Kennzahl in einer Tabelle. *Die Funktion Column() berücksichtigt die Dimensionsspalten nicht.*

Beschränkungen:

- Die Rekursion liefert NULL.
- Falls **ColumnNo** auf eine Spalte referenziert, für die keine Kennzahl vorhanden ist, wird der Wert NULL zurückgegeben.
- Das Sortieren nach y-Werten in Diagrammen oder nach Formelspalten in Tabellen ist nicht zulässig, wenn diese Diagrammfunktion in einer der Diagrammformeln verwendet wird. Diese Sortierungsoptionen werden daher automatisch deaktiviert. Wenn Sie diese Diagrammfunktion in einer Visualisierung oder Tabelle verwenden, wird die Sortierung der Visualisierung auf die sortierte Eingabe dieser Funktion zurückgesetzt.

Beispiele und Ergebnisse:

Beispiel: Prozentsatz Gesamtumsatz

Customer	Product	UnitPrice	UnitSales	Order Value	Total Sales Value	% Sales
A	AA	15	10	150	505	29.70
A	AA	16	4	64	505	12.67
A	BB	9	9	81	505	16.04

Customer	Product	UnitPrice	UnitSales	Order Value	Total Sales Value	% Sales
B	BB	10	5	50	505	9.90
B	CC	20	2	40	505	7.92
B	DD	25	-	0	505	0.00
C	AA	15	8	120	505	23.76
C	CC	19	-	0	505	0.00

Beispiel: Prozentsatz Umsatz für ausgewählten Kunden

Customer	Product	UnitPrice	UnitSales	Order Value	Total Sales Value	% Sales
A	AA	15	10	150	295	50.85
A	AA	16	4	64	295	21.69
A	BB	9	9	81	295	27.46

Beispiele und Ergebnisse

Beispiele	Ergebnisse
Order Value wird zur Tabelle als Kennzahl mit der folgenden Formel hinzugefügt: `Sum (UnitPrice*UnitSales)`. Total Sales Value wird als Kennzahl mit der folgenden Formel hinzugefügt: `Sum(TOTAL UnitPrice*UnitSales)` % Sales wird als Kennzahl mit der folgenden Formel hinzugefügt: `100*Column(1)/Column (2)`	Das Ergebnis von Column(1) wird der Spalte Order Value entnommen, weil es sich hierbei um die erste Kennzahlspalte handelt. Das Ergebnis von Column(2) wird der Spalte Total Sales Value entnommen, weil es sich hierbei um die zweite Kennzahlspalte handelt. Siehe die Ergebnisse in der Spalte % Sales des Beispiels *Prozentsatz Gesamtumsatz (page 1316)*.
Treffen Sie die Auswahl Customer A.	Diese Auswahl ändert Total Sales Value und zugleich %Sales. Siehe das Beispiel *Prozentsatz Umsatz für ausgewählten Kunden (page 1317)*.

In Beispielen verwendete Daten:

```
ProductData:
LOAD * inline [
Customer|Product|UnitSales|UnitPrice
Astrida|AA|4|16
Astrida|AA|10|15
Astrida|BB|9|9
Betacab|BB|5|10
Betacab|CC|2|20
Betacab|DD||25
```

```
Canutility|AA|8|15
Canutility|CC||19
] (delimiter is '|');
```

Dimensionality - Diagrammfunktion

Dimensionality() liefert für die aktuelle Zeile die Anzahl der Dimensionen. Im Fall von Pivottabellen liefert die Funktion die Gesamtzahl der Dimensionsspalten ohne aggregierten Inhalt, d. h. ohne Partialsummen und ausgeblendete Dimensionen.

Syntax:

```
Dimensionality ( )
```

Rückgabe Datentyp: ganze Zahl

Beschränkungen:

Diese Funktion ist nur in Diagrammen verfügbar. Für alle Diagrammtypen außer Pivottabellen entspricht das Ergebnis der Zahl der Dimensionen in allen Zeilen außer der Summenzeile, dort wird der Wert 0 zurückgegeben.

Das Sortieren nach y-Werten in Diagrammen oder nach Formelspalten in Tabellen ist nicht zulässig, wenn diese Diagrammfunktion in einer der Diagrammformeln verwendet wird. Diese Sortierungsoptionen werden daher automatisch deaktiviert. Wenn Sie diese Diagrammfunktion in einer Visualisierung oder Tabelle verwenden, wird die Sortierung der Visualisierung auf die sortierte Eingabe dieser Funktion zurückgesetzt.

Beispiel: Diagrammformel mit Dimensionalität

Beispiel: Diagrammformel
Die Funktion **Dimensionality()** kann mit einer Pivottabelle als Diagrammformel verwendet werden, wenn Sie unterschiedliche Zellenformatierung abhängig von der Anzahl der Dimensionen einer Zeile mit nicht aggregierten Daten anwenden möchten. In diesem Beispiel wird die Funktion Dimensionality() verwendet, um eine Hintergrundfarbe auf Tabellenzellen anzuwenden, die einer bestimmten Bedingung entsprechen.

Ladeskript

Laden Sie die folgenden Daten als Inline-Ladevorgang in den Dateneditor, um das folgende Diagrammformelbeispiel zu erstellen.

```
ProductSales:

Load * inline [
Country,Product,Sales,Budget
Sweden,AA,100000,50000
Germany,AA,125000,175000
Canada,AA,105000,98000
Norway,AA,74850,68500
Ireland,AA,49000,48000
Sweden,BB,98000,99000
Germany,BB,115000,175000
Norway,BB,71850,68500
Ireland,BB,31000,48000
] (delimiter is ',');
```

Diagrammformel

Erstellen Sie eine Pivottabellenvisualisierung in einem Qlik Sense Arbeitsblatt mit **Country** und **Product** als Dimensionen. Fügen Sie **Sum(Sales)**, **Sum(Budget)** und **Dimensionality()** als Kennzahlen hinzu.

Geben Sie im Fenster **Eigenschaften** die folgende Formel als **Formel für die Hintergrundfarbe** für die Kennzahl **Sum(Sales)** ein.

```
If(Dimensionality()=1 and Sum(Sales)<Sum(Budget),RGB(255,156,156),
If(Dimensionality()=2 and Sum(Sales)<Sum(Budget),RGB(178,29,29)
))
```

Ergebnis:

Country / Product | Values

	Sum(Sales)	Sum([Budget])	Dimensionality()
Canada	105000	98000	1
AA	105000	98000	2
Germany	240000	350000	1
Ireland	80000	96000	1
AA	49000	48000	2
BB	31000	48000	2
Norway	146700	137000	1
AA	74850	68500	2
BB	71850	68500	2
Sweden	198000	149000	1

Erläuterung

Die Formel `If(Dimensionality()=1 and Sum(Sales)<Sum(Budget),RGB(255,156,156), If(Dimensionality()=2 and Sum(Sales)<Sum(Budget),RGB(178,29,29)))` enthält bedingte Anweisungen, die den Dimensionalitätswert und „Sum(Sales)“ und „Sum(Budget)“ für jedes Produkt prüfen. Wenn die Bedingungen erfüllt sind, wird eine Hintergrundfarbe auf den Wert „Sum(Sales)“ angewendet.

Exists

Exists() bestimmt, ob ein spezifischer Feldwert bereits in das Feld im Datenladeskript geladen wurde. Die Funktion gibt TRUE oder FALSE zurück, und kann deshalb in der **where**-Bedingung eines **LOAD**-Befehls oder eines **IF**-Befehls verwendet werden.

*Sie können auch **Not Exists()** verwenden, um zu bestimmen, ob ein Feldwert nicht geladen wurde. Bei der Verwendung von **Not Exists()** in einer where-Bedingung wird allerdings zu Vorsicht geraten. Mit der **Exists()**-Funktion werden sowohl zuvor geladene Tabellen als auch zuvor geladene Werte in der aktuellen Tabelle geprüft. Daher wird nur das erste Vorkommen geladen. Wenn das zweite Vorkommen gefunden wird, ist der Wert bereits geladen. In den Beispielen finden Sie weitere Informationen.*

Syntax:

```
Exists(field_name [, expr])
```

Rückgabe Datentyp: Boolesch

Argumente:

Argumente

Argument	Beschreibung
field_name	Der Name des Felds, in dem Sie nach einem Wert suchen möchten. Sie können einen expliziten Feldnamen ohne Anführungszeichen verwenden. Das Feld muss bereits vom Skript geladen worden sein. Das bedeutet, dass Sie sich nicht auf ein Feld beziehen können, das von einer Bedingung an späterer Stelle im Skript geladen wird.
expr	Der Wert, dessen Vorhandensein Sie prüfen möchten. Sie können einen expliziten Wert oder eine Formel verwenden, die sich auf ein oder mehrere Felder in der aktuellen load-Anweisung bezieht. *Sie können sich nicht auf Felder beziehen, die nicht in der aktuellen load-Anweisung enthalten sind.* Dieses Argument ist optional. Wenn Sie es auslassen, wird mit der Funktion geprüft, ob der Wert von **field_name** im aktuellen Datensatz bereits vorhanden ist.

Beispiele und Ergebnisse:

Beispiel 1

```
Exists (Employee)
```

Liefert -1 (True), wenn der Feldwert **Employee** im aktuellen Datensatz bereits in einem anderen Datensatz in diesem Feld vorkommt.

Die Befehle `Exists (Employee, Employee)` und `Exists (Employee)` sind gleichwertig.

Beispiel 2

```
Exists(Employee, 'Bill')
```

Liefert -1 (True), wenn der Feldwert **'Bill'** im aktuellen Inhalt von Feld **Employee** gefunden wird.

Beispiel 3

```
Employees:
LOAD * inline [
Employee|ID|Salary
Bill|001|20000
John|002|30000
Steve|003|35000
] (delimiter is '|');

Citizens:
Load * inline [
Employee|Address
Bill|New York
Mary|London
Steve|Chicago
Lucy|Madrid
Lucy|Paris
John|Miami
] (delimiter is '|') where Exists (Employee);

Drop Tables Employees;
```

Dies führt zu einer Tabelle, die Sie in einer Tabellenvisualisierung mit den Dimensionen Employee und Address verwenden können.

Die Bedingung where (`where Exists (Employee)`) bedeutet, dass nur Namen aus der Tabelle Citizens, die sich auch in Employees befinden, in die neue Tabelle geladen werden. Der Drop-Befehl entfernt die Tabelle Employees, um Verwechslungen zu vermeiden.

Ergebnisse

Employee	Address
Bill	New York
John	Miami
Steve	Chicago

Beispiel 4

```
Employees:
Load * inline [
Employee|ID|Salary
Bill|001|20000
```

```
John|002|30000
Steve|003|35000
] (delimiter is '|');

Citizens:
Load * inline [
Employee|Address
Bill|New York
Mary|London
Steve|Chicago
Lucy|Madrid
Lucy|Paris
John|Miami
] (delimiter is '|') where not Exists (Employee);

Drop Tables Employees;
```

Die Bedingung where beinhaltet not: `where not Exists (Employee)`.

Das bedeutet, dass nur Namen aus der Tabelle Citizens, die sich nicht in Employees befinden, in die neue Tabelle geladen werden.

Beachten Sie, dass für Lucy zwei Werte in der Tabelle Citizens vorhanden sind, jedoch nur einer davon in der Ergebnistabelle enthalten ist. Wenn Sie die erste Zeile mit dem Wert Lucy laden, wird der Wert in das Feld Employee aufgenommen. Wenn die zweite Zeile geprüft wird, ist daher der Wert bereits vorhanden.

Ergebnisse

Employee	**Adresse**
Mary	London
Lucy	Madrid

Beispiel 5

In diesem Beispiel wird gezeigt, wie Sie alle Werte laden.

```
Employees:
Load Employee As Name;
LOAD * inline [
Employee|ID|Salary
Bill|001|20000
John|002|30000
Steve|003|35000
] (delimiter is '|');

Citizens:
Load * inline [
Employee|Address
Bill|New York
Mary|London
Steve|Chicago
Lucy|Madrid
```

```
Lucy|Paris
John|Miami
] (delimiter is '|') where not Exists (Name, Employee);

Drop Tables Employees;
```

Damit alle Werte für Lucy abgerufen werden, müssen Sie zwei Einstellungen ändern:

- Ein vorausgehender Ladevorgang in die Tabelle Employees wurde eingefügt, wobei Employee in Name umbenannt wurde.
  `Load Employee As Name;`
- Die Where-Bedingung in Citizens wurde wie folgt geändert:
  `not Exists (Name, Employee).`

Damit werden Felder für Name und Employee erstellt. Wenn die zweite Zeile mit Lucy geprüft wird, ist sie in Name noch nicht vorhanden.

Ergebnisse

Employee	**Adresse**
Mary	London
Lucy	Madrid
Lucy	Paris

FieldIndex

FieldIndex() liefert die Position des Wertes **value** im Feld **field_name** (nach Lade-Reihenfolge).

Syntax:

```
FieldIndex(field_name , value)
```

Rückgabe Datentyp: ganze Zahl

Argumente:

Argumente

Argument	**Beschreibung**
field_name	Name für das Feld, für das der Index erforderlich ist. Zum Beispiel die Spalte in einer Tabelle. Muss als String angegeben werden. Das heißt, der Feldname muss in einfachen Anführungszeichen stehen.
value	Der Wert des Feldes **field_name**.

Beschränkungen:

- Ist **value** kein Wert des Feldes **field_name**, ist das Ergebnis 0.
- Das Sortieren nach y-Werten in Diagrammen oder nach Formelspalten in Tabellen ist nicht zulässig, wenn diese Diagrammfunktion in einer der Diagrammformeln verwendet wird. Diese Sortierungsoptionen werden daher automatisch deaktiviert. Wenn Sie diese Diagrammfunktion in einer Visualisierung oder Tabelle verwenden, wird die Sortierung der Visualisierung auf die sortierte Eingabe dieser Funktion zurückgesetzt. Diese Einschränkung gilt nicht für die entsprechende Skriptfunktion.

Beispiele und Ergebnisse:

Die folgenden Beispiele verwenden das Feld: **First name** aus der Tabelle **Names**.

Beispiele und Ergebnisse

Beispiele	**Ergebnisse**
Fügen Sie Ihrer App die Beispieldaten hinzu und führen Sie sie aus.	Die Tabelle **Names** wird wie in der Datenstichprobe geladen.
Diagrammfunktion: Fügen Sie einer Tabelle mit der Dimension First name die folgende Kennzahl hinzu:	
`FieldIndex ('First name','John')`	1, weil 'John' zuerst in der Lade-Reihenfolge des Feldes **First name** aufgeführt wird. Hinweis: In einem Filterfenster würde **John** als 2. von oben angezeigt werden, da alphabetisch und nicht nach Lade-Reihenfolge sortiert wird.
`FieldIndex ('First name','Peter')`	4, weil **FieldIndex()** nur einen Wert zurückgibt, d. h. das erste Auftreten in der Lade-Reihenfolge.
Skriptfunktion: Vorausgesetzt, die Tabelle **Names** ist wie in den Beispieldaten geladen:	

`John1:`
`Load FieldIndex('First name','John') as MyJohnPos`
`Resident Names;`
	`MyJohnPos=1`, weil 'John' zuerst in der Lade-Reihenfolge des Feldes **First name** aufgeführt wird. Hinweis: In einem Filterfenster würde **John** als 2. von oben angezeigt werden, da alphabetisch und nicht nach Lade-Reihenfolge sortiert wird.

`Peter1:`
`Load FieldIndex('First name','Peter') as MyPeterPos`
`Resident Names;`
	`MyPeterPos=4`, weil **FieldIndex()** nur einen Wert liefert, d. h. das erste Auftreten in der Lade-Reihenfolge.

Im Beispiel verwendete Daten:

```
Names:
LOAD * inline [
First name|Last name|Initials|Has cellphone
John|Anderson|JA|Yes
Sue|Brown|SB|Yes
```

```
Mark|Carr|MC|No
Peter|Devonshire|PD|No
Jane|Elliot|JE|Yes
Peter|Franc|PF|Yes ] (delimiter is '|');

John1:
Load FieldIndex('First name','John') as MyJohnPos
Resident Names;

Peter1:
Load FieldIndex('First name','Peter') as MyPeterPos
Resident Names;
```

FieldValue

FieldValue() liefert den Wert an der (nach Lade-Reihenfolge) **elem_no**-ten Position des Feldes **field_name**.

Syntax:

```
FieldValue(field_name , elem_no)
```

Rückgabe Datentyp: dual

Argumente:

Argumente

Argument	Beschreibung
field_name	Name für das Feld, für das der Wert erforderlich ist. Zum Beispiel die Spalte in einer Tabelle. Muss als String angegeben werden. Das heißt, der Feldname muss in einfachen Anführungszeichen stehen.
elem_no	Die Positionsnummer (Element) des Feldes gemäß der Lade-Reihenfolge, für die der Wert ausgegeben wird. Dies könnte einer Zeile in einer Tabelle entsprechen, hängt aber davon ab, in welcher Reihenfolge die Elemente (Zeilen) geladen werden.

Beschränkungen:

- Ist **elem_no** größer als die Zahl der Feldwerte, ist das Ergebnis NULL.
- Das Sortieren nach y-Werten in Diagrammen oder nach Formelspalten in Tabellen ist nicht zulässig, wenn diese Diagrammfunktion in einer der Diagrammformeln verwendet wird. Diese Sortierungsoptionen werden daher automatisch deaktiviert. Wenn Sie diese Diagrammfunktion in einer Visualisierung oder Tabelle verwenden, wird die Sortierung der Visualisierung auf die sortierte Eingabe dieser Funktion zurückgesetzt. Diese Einschränkung gilt nicht für die entsprechende Skriptfunktion.

Beispiel

Ladeskript

Laden Sie die folgenden Daten als Inline-Ladevorgang in den Dateneditor, um das folgende Beispiel zu erstellen.

```
Names:

LOAD * inline [
First name|Last name|Initials|Has cellphone
John|Anderson|JA|Yes
Sue|Brown|SB|Yes
Mark|Carr|MC |No
Peter|Devonshire|PD|No
Jane|Elliot|JE|Yes
Peter|Franc|PF|Yes ] (delimiter is '|');

John1:

Load FieldValue('First name',1) as MyPos1
Resident Names;

Peter1:

Load FieldValue('First name',5) as MyPos2
Resident Names;
```

Erstellen einer Visualisierung

Erstellen Sie eine Tabellenvisualisierung in einem Qlik Sense-Arbeitsblatt. Fügen Sie die Felder **First name**, **MyPos1** und **MyPos2** zur Tabelle hinzu.

Ergebnis

First name	MyPos1	MyPos2
Jane	John	Jane
John	John	Jane
Mark	John	Jane
Peter	John	Jane
Sue	John	Jane

Erläuterung

FieldValue('First name','1') ergibt „John“ als Wert für **MyPos1** für alle Vornamen, weil „John“ in der Ladereihenfolge des Felds **First name** an erster Stelle steht. Hinweis: In einem Filterfenster würde John als zweiter von oben nach Jane angezeigt werden, da alphabetisch und nicht nach Lade-Reihenfolge sortiert wird.

FieldValue('First name','5') ergibt „Jane“ als Wert für **MyPos2** für alle Vornamen, weil „Jane“ in der Ladereihenfolge des Felds **First name** an fünfter Stelle steht.

FieldValueCount

FieldValueCount() ist eine **integer**-Funktion, die die Anzahl der distinkten Werte in einem Feld zurückgibt.

Mit einem partiellen Ladevorgang können Werte aus den Daten entfernt werden, die nicht in der zurückgegebenen Zahl enthalten sind. Die zurückgegebene Zahl entspricht allen distinkten Werten, die entweder beim anfänglichen Ladevorgang oder bei allen anschließenden partiellen Ladevorgängen geladen wurden.

Das Sortieren nach y-Werten in Diagrammen oder nach Formelspalten in Tabellen ist nicht zulässig, wenn diese Diagrammfunktion in einer der Diagrammformeln verwendet wird. Diese Sortierungsoptionen werden daher automatisch deaktiviert. Wenn Sie diese Diagrammfunktion in einer Visualisierung oder Tabelle verwenden, wird die Sortierung der Visualisierung auf die sortierte Eingabe dieser Funktion zurückgesetzt. Diese Einschränkung gilt nicht für die entsprechende Skriptfunktion.

Syntax:

```
FieldValueCount(field_name)
```

Rückgabe Datentyp: ganze Zahl

Argumente:

Argumente

Argument	Beschreibung
field_name	Name für das Feld, für das der Wert erforderlich ist. Zum Beispiel die Spalte in einer Tabelle. Muss als String angegeben werden. Das heißt, der Feldname muss in einfachen Anführungszeichen stehen.

Beispiele und Ergebnisse:

Die folgenden Beispiele verwenden das Feld **First name** aus der Tabelle **Names**.

Beispiele und Ergebnisse

Beispiele	Ergebnisse
Fügen Sie Ihrer App die Beispieldaten hinzu und führen Sie sie aus.	Die Tabelle **Names** wird wie in der Datenstichprobe geladen.
Diagrammfunktion: Fügen Sie einer Tabelle mit der Dimension First name Folgendes als Kennzahl hinzu:	
`FieldValueCount('First name')`	5, da **Peter** doppelt vorkommt.
`FieldValueCount('Initials')`	6, da **Initials** nur distinkte Werte aufweist.
Skriptfunktion: Vorausgesetzt, die Tabelle **Names** ist wie in den Beispieldaten geladen:	

Beispiele	Ergebnisse
FieldCount1: Load FieldValueCount('First name') as MyFieldCount1 Resident Names;	MyFieldCount1=5, weil Peter doppelt vorkommt.
FieldCount2: Load FieldValueCount('Initials') as MyInitialsCount1 Resident Names;	MyFieldCount1=6, weil 'Initials' nur distinkte Werte aufweist.

In Beispielen verwendete Daten:

```
Names:
LOAD * inline [
First name|Last name|Initials|Has cellphone
John|Anderson|JA|Yes
Sue|Brown|SB|Yes
Mark|Carr|MC|No
Peter|Devonshire|PD|No
Jane|Elliot|JE|Yes
Peter|Franc|PF|Yes ] (delimiter is '|');

FieldCount1:
Load FieldValueCount('First name') as MyFieldCount1
Resident Names;

FieldCount2:
Load FieldValueCount('Initials') as MyInitialsCount1
Resident Names;
```

LookUp

Lookup() sucht in einer bereits geladenen Tabelle und liefert den Wert des Feldes **field_name**, der dem ersten Auftreten des Werts **match_field_value** im Feld **match_field_name** zugehörig ist. Bei der Tabelle kann es sich um die aktuelle Tabelle oder eine andere zuvor geladene Tabelle handeln.

Syntax:

```
lookup(field_name, match_field_name, match_field_value [, table_name])
```

Rückgabe Datentyp: dual

Argumente:

Argumente

Argument	Beschreibung
field_name	Name für das Feld, für das der Rückgabewert erforderlich ist. Eingabewert muss als String angegeben werden (zum Beispiel in einfachen Anführungszeichen).
match_field_name	Name des Felds, in dem **match_field_value** aufgerufen werden soll. Eingabewert muss als String angegeben werden (zum Beispiel in einfachen Anführungszeichen).

Argument	Beschreibung
match_field_value	Wert, der im Feld **match_field_name** aufgerufen werden soll.
table_name	Name der Tabelle, in welcher der Wert aufgerufen werden soll. Eingabewert muss als String angegeben werden (zum Beispiel in einfachen Anführungszeichen). Ist **table_name** nicht angegeben, sucht das Programm in der aktuellen Tabelle.

Argumente ohne Anführungszeichen beziehen sich auf die aktuelle Tabelle. Um den Bezug zu anderen Tabellen herzustellen, muss ein Argument in einzelnen Anführungszeichen stehen.

Beschränkungen:

Die Reihenfolge, in der die Suche erfolgt, entspricht der Lade-Reihenfolge, sofern die Tabelle nicht durch komplexe Operationen wie etwa Joins entsteht. **field_name** und **match_field_name** müssen beide Felder derselben Tabelle **table_name** sein.

Wird keine Übereinstimmung gefunden, ist das Ergebnis NULL.

Beispiel

Ladeskript

Laden Sie die folgenden Daten als Inline-Ladevorgang in den Dateneditor, um das folgende Beispiel zu erstellen.

```
ProductList:
Load * Inline [
ProductID|Product|Category|Price
1|AA|1|1
2|BB|1|3
3|CC|2|8
4|DD|3|2
] (delimiter is '|');

OrderData:
Load *, Lookup('Category', 'ProductID', ProductID, 'ProductList') as CategoryID
Inline [
InvoiceID|CustomerID|ProductID|Units
1|Astrida|1|8
1|Astrida|2|6
2|Betacab|3|10
3|Divadip|3|5
4|Divadip|4|10
] (delimiter is '|');

Drop Table ProductList;
```

Erstellen einer Visualisierung

Erstellen Sie eine Tabellenvisualisierung in einem Qlik Sense-Arbeitsblatt. Fügen Sie die Felder **ProductID**, **InvoiceID**, **CustomerID**, **Units** und **CategoryID** zur Tabelle hinzu.

Ergebnis

Ergebnistabelle

ProductID	InvoiceID	CustomerID	Einheiten	CategoryID
1	1	Astrida	8	1
2	1	Astrida	6	1
3	2	Betacab	10	2
3	3	Divadip	5	2
4	4	Divadip	10	3

Erläuterung

Die Beispieldaten verwenden die Funktion **Lookup()** in der folgenden Form:

```
Lookup('Category', 'ProductID', ProductID, 'ProductList')
```

Die Tabelle **ProductList** wird zuerst geladen.

Die Funktion **Lookup()** wird zum Erstellen der Tabelle **OrderData** verwendet. Sie legt das dritte Argument als **ProductID** fest. Das ist das Feld, für das der Wert im zweiten Argument **'ProductID'** in der **ProductList** gesucht wird, wie durch die einfachen Anführungszeichen angegeben wurde.

Die Funktion liefert den Wert für '**Category**' (in der Tabelle **ProductList**), geladen als **CategoryID**.

Der Befehl **drop** löscht die Tabelle **ProductList** aus dem Datenmodell, weil sie nicht erforderlich ist, wodurch die Tabelle **OrderData** zurückbleibt.

Die Funktion Lookup() ist flexibel und kann auf zuvor geladene Tabellen zugreifen. Im Vergleich zur Funktion Applymap() ist sie jedoch langsam.

Siehe auch:

ApplyMap (page 1353)

NoOfRows - Diagrammfunktion

NoOfRows() liefert die Anzahl der Zeilen im aktuellen Spaltenabschnitt in einer Tabelle. In Bitmap-Diagrammen liefert **NoOfRows()** die Zahl der Zeilen im entsprechenden Äquivalent zum Tabellendiagramm.

Hat das Diagramm dagegen mehrere vertikale Dimensionen, so umfasst der Spaltenabschnitt nur Zeilen, deren Werte in allen Dimensionen mit Ausnahme der in der Priorität der Sortierfolgen letzten Dimension übereinstimmen.

Das Sortieren nach y-Werten in Diagrammen oder nach Formelspalten in Tabellen ist nicht zulässig, wenn diese Diagrammfunktion in einer der Diagrammformeln verwendet wird. Diese Sortierungsoptionen werden daher automatisch deaktiviert. Wenn Sie diese Diagrammfunktion in einer Visualisierung oder Tabelle verwenden, wird die Sortierung der Visualisierung auf die sortierte Eingabe dieser Funktion zurückgesetzt.

Syntax:

```
NoOfRows([TOTAL])
```

Rückgabe Datentyp: ganze Zahl

Argumente:

Argumente

Argument	Beschreibung
TOTAL	Wenn das Diagramm nur eine Dimension hat oder die Formel mit dem Zusatz **TOTAL** als Argument versehen ist, entspricht der Spaltenabschnitt der gesamten Spalte.

Beispiel: Diagrammformel mit NoOfRows

Beispiel – Diagrammformel

Ladeskript

Laden Sie die folgenden Daten als Inline-Ladevorgang in den Dateneditor, um die folgenden Diagrammformelbeispiele zu erstellen.

```
        Temp:
LOAD * inline [
Region|SubRegion|RowNo()|NoOfRows()
Africa|Eastern
Africa|Western
Americas|Central
Americas|Northern
Asia|Eastern
Europe|Eastern
Europe|Northern
Europe|Western
Oceania|Australia
] (delimiter is '|');
```

Diagrammformel

Erstellen Sie eine Tabellenvisualisierung in einem Qlik Sense Arbeitsblatt mit **Region** und **SubRegion** als Dimensionen. Fügen Sie `RowNo( )`, `NoOfRows()` und `NoOfRows(Total)` als Kennzahlen hinzu.

Ergebnis

Region	SubRegion	RowNo()	NoOfRows()	NoOfRows (Total)
Africa	Eastern	1	2	9
Africa	Western	2	2	9
Americas	Central	1	2	9
Americas	Northern	2	2	9
Asia	Eastern	1	1	9
Europe	Eastern	1	3	9
Europe	Northern	2	3	9
Eurrope	Western	3	3	9
Oceania	Australia	1	1	9

Erläuterung

In diesem Beispiel ist die Sortierreihenfolge nach der ersten Dimension Region. Daher setzt sich jedes Spaltensegment aus einer Gruppe von Regionen zusammen, die den gleichen Wert haben, z. B. Afrika.

Die Spalte **RowNo()** zeigt die Zeilenanzahl für jedes Spaltensegment. Beispielsweise sind zwei Zeilen für die Region Afrika vorhanden. Die Zeilennummerierung beginnt dann für den nächsten Spaltenabschnitt, Americas, wieder bei 1.

Die Spalte **NoOfRows()** zählt die Zeilenanzahl in jedem Spaltensegment. Zum Beispiel hat Europa drei Zeilen im Spaltensegment.

Die Spalte **NoOfRows(Total)** berücksichtigt die Dimensionen aufgrund des Arguments TOTAL für NoOfRows() nicht und zählt die Zeilen in der Tabelle.

Wenn die Tabelle nach der zweiten Dimension, SubRegion, sortiert wäre, würden die Spaltensegmente auf dieser Dimension basieren, so dass sich die Zeilenanzahl für jede SubRegion ändern würde.

Siehe auch:

RowNo - Diagrammfunktion (page 607)

Peek

Peek() gibt den Wert eines Feldes in einer Tabelle für eine Zeile zurück, die bereits geladen wurde. Die Zeilennummer kann wie die Tabelle festgelegt werden. Wenn keine Zeilennummer angegeben ist, wird der letzte zuvor geladene Datensatz verwendet.

Die Funktion peek() wird am häufigsten verwendet, um die relevanten Grenzwerte in einer zuvor geladenen Tabelle zu finden, also den ersten Wert bzw. den letzten Wert eines bestimmten Feldes. In den meisten Fällen wird dieser Wert zur späteren Verwendung in einer Variablen gespeichert, beispielsweise als Bedingung in einer do-while-Schleife.

Syntax:

```
Peek (

field_name

[, row_no[, table_name ] ])
```

Rückgabe Datentyp: dual

Argumente:

Argumente

Argument	Beschreibung
field_name	Name für das Feld, für das der Rückgabewert erforderlich ist.Eingabewert muss als String angegeben werden (zum Beispiel in einfachen Anführungszeichen).
row_no	Die Zeile in der Tabelle, die das erforderliche Feld enthält. Kann eine Formel sein, die aber eine ganze Zahl ergeben muss. 0 steht für den ersten Datensatz, 1 für den zweiten usw. Mithilfe von negativen Zahlen können die Datensätze vom unteren Ende der Tabelle aus gezählt werden. -1 bezeichnet den letzten gelesenen Datensatz. Fehlt **row_no**, wird -1 angenommen.
table_name	Ein Tabellenname ohne abschließenden Doppelpunkt. Fehlt **table_name**, wird die aktuelle Tabelle verwendet. Wird die Funktion außerhalb des **LOAD**-Befehls verwendet oder bezieht sie sich auf eine andere Tabelle, muss **table_name**explizit angegeben werden.

Beschränkungen:

Die Funktion kann nur Werte aus bereits geladenen Datensätzen zurückgeben. Das bedeutet, dass im ersten Datensatz einer Tabelle ein Aufruf, der -1 als row_no verwendet, NULL zurückgibt.

Beispiele und Ergebnisse:

Beispiel 1

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Fügen Sie einem Arbeitsblatt in Ihrer App dann die Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um das Ergebnis anzuzeigen.

```
EmployeeDates:
Load * Inline [
EmployeeCode|StartDate|EndDate
101|02/11/2010|23/06/2012
102|01/11/2011|30/11/2013
103|02/01/2012|
```

```
104|02/01/2012|31/03/2012
105|01/04/2012|31/01/2013
106|02/11/2013|
] (delimiter is '|');

First_last_Employee:
Load
EmployeeCode,
Peek('EmployeeCode',0,'EmployeeDates') As FirstCode,
Peek('EmployeeCode',-1,'EmployeeDates') As LastCode
Resident EmployeeDates;
```

Ergebnistabelle

Mitarbeitercode	StartDate	EndDate	FirstCode	LastCode
101	02/11/2010	23/06/2012	101	106
102	01/11/2011	30/11/2013	101	106
103	02/01/2012		101	106
104	02/01/2012	31/03/2012	101	106
105	01/04/2012	31/01/2013	101	106
106	02/11/2013		101	106

FirstCode = 101, weil `Peek('EmployeeCode',0, 'EmployeeDates')` den ersten Wert von EmployeeCode in der Tabelle EmployeeDates liefert.

LastCode = 106, weil `Peek('EmployeeCode',-1, 'EmployeeDates')` den letzten Wert von EmployeeCode in der Tabelle EmployeeDates liefert.

Durch den Ersatz des Werts des Arguments **row_no** werden die Werte von anderen Zeilen in der Tabelle wie nachfolgend beschrieben ausgegeben:

`Peek('EmployeeCode',2, 'EmployeeDates')` liefert den dritten Wert (103) in der Tabelle als den FirstCode.

Beachten Sie jedoch, dass sich die Funktion ohne Festlegen der Tabelle als drittes Argument **table_name** in diesen Beispielen auf die aktuelle (in diesem Fall die interne) Tabelle bezieht.

Beispiel 2

Wenn Sie auf Daten weiter unten in einer Tabelle zugreifen möchten, müssen Sie zwei Schritte durchführen: Laden Sie zuerst die ganze Tabelle in eine temporäre Tabelle und sortieren Sie diese dann anhand von **Peek()** neu.

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Fügen Sie einem Arbeitsblatt in Ihrer App dann die Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um das Ergebnis anzuzeigen.

```
T1:
LOAD * inline [
ID|Value
1|3
1|4
1|6
```

```
3|7
3|8
2|1
2|11
5|2
5|78
5|13
] (delimiter is '|');

T2:

LOAD *,
IF(ID=Peek('ID'), Peek('List')&','&Value,Value) AS List
RESIDENT T1
ORDER BY ID ASC;
DROP TABLE T1;
```

Create a table in a sheet in your app with **ID**, **List**, and **Value** as the dimensions.

Ergebnistabelle

ID	Liste	Wert
1	3,4	4
1	3,4,6	6
1	3	3
2	1,11	11
2	1	1
3	7,8	8
3	7	7
5	2,78	78
5	2,78,13	13
5	2	2

Der Befehl **IF()** wird über die temporäre Tabelle T1 ausgeführt.
`Peek('ID')` bezieht sich auf das Feld ID in der vorhergehenden Zeile in der aktuellen Tabelle T2.
`Peek('List')` bezieht sich auf das Feld List in der vorhergehenden Zeile in der Tabelle T2, die derzeit erstellt wird, während die Formel überprüft wird.

Der Befehl liefert folgendes Ergebnis:
Wenn der aktuelle Wert von ID mit dem vorhergehenden Wert von ID übereinstimmt, wird der Wert von Peek ('List') mit dem aktuellen Wert von Value zusammengefasst. Ansonsten wird nur der aktuelle Wert von Value geschrieben.

Enthält Peek('List') bereits ein zusammengefasstes Ergebnis, wird das neue Ergebnis von Peek('List') damit zusammengefasst.

*Beachten Sie die Bedingung **Order by**. Diese gibt an, wie die Tabelle sortiert wird (nach ID in aufsteigender Reihenfolge). Ohne diese Bedingung nutzt die Funktion Peek() die beliebige Sortierung der internen Tabelle, was zu unvorhersehbaren Ergebnissen führen kann.*

Beispiel 3

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Fügen Sie einem Arbeitsblatt in Ihrer App dann die Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um das Ergebnis anzuzeigen.

```
Amounts:
Load
Date#(Month,'YYYY-MM') as Month,
Amount,
Peek(Amount) as AmountMonthBefore
Inline
[Month,Amount
2022-01,2
2022-02,3
2022-03,7
2022-04,9
2022-05,4
2022-06,1];
```

Ergebnistabelle

Amount	AmountMonthBefore	Monat
1	4	2022-06
2	-	2022-01
3	2	2022-02
4	9	2022-05
7	3	2022-03
9	7	2022-04

Das Feld AmountMonthBefore enthält den Betrag aus dem Vormonat.

Hier werden die Parameter row_no und table_name ausgelassen, sodass die Standardwerte verwendet werden. In diesem Beispiel sind die folgenden drei Funktionsaufrufe äquivalent.

- Peek(Amount)
- Peek(Amount,-1)
- Peek(Amount,-1,'Amounts')

Wenn -1 als row_no verwendet wird, bedeutet dies, dass der Wert aus der vorigen Zeile verwendet wird. Durch Ersetzen dieses Werts können Werte aus anderen Zeilen in der Tabelle abgerufen werden:

Peek(Amount,2) gibt den dritten Wert in der Tabelle zurück: 7.

Beispiel 4

Die Daten müssen korrekt sortiert sein, um das korrekte Ergebnis zu erhalten. Leider ist dies nicht immer der Fall. Zudem kann die Funktion Peek() nicht zum Verweisen auf noch nicht geladene Daten verwendet werden. Mit temporären Tabellen und der Ausführung mehrerer Durchläufe durch die Daten lassen sich derartige Probleme vermeiden.

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Fügen Sie einem Arbeitsblatt in Ihrer App dann die Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um das Ergebnis anzuzeigen.

```
tmp1Amounts:
Load * Inline
[Month,Product,Amount
2022-01,B,3
2022-01,A,8
2022-02,B,4
2022-02,A,6
2022-03,B,1
2022-03,A,6
2022-04,A,5
2022-04,B,5
2022-05,B,6
2022-05,A,7
2022-06,A,4
2022-06,B,8];

tmp2Amounts:
Load *,
If(Product=Peek(Product),Peek(Amount)) as AmountMonthBefore
Resident tmp1Amounts
Order By Product, Month Asc;
Drop Table tmp1Amounts;

Amounts:
Load *,
If(Product=Peek(Product),Peek(Amount)) as AmountMonthAfter
Resident tmp2Amounts
Order By Product, Month Desc;
Drop Table tmp2Amounts;
```

Erläuterung

Die anfängliche Tabelle ist nach Monat sortiert, was bedeutet, dass die Funktion peek() in vielen Fällen den Betrag für das falsche Produkt zurückgeben würde. Daher muss diese Tabelle neu sortiert werden. Das geschieht durch einen zweiten Durchlauf durch die Daten, mit dem eine neue Tabelle namens „tmp2Amounts“ erstellt wird. Beachten Sie die Bedingung „Order by“. Damit werden die Datensätze zuerst nach Produkt und dann nach Monat in aufsteigender Reihenfolge sortiert.

Die Funktion If() wird benötigt, da AmountMonthBefore nur berechnet werden sollte, wenn die vorherige Zeile die Daten für das gleiche Produkt, aber den Vormonat enthält. Durch Vergleichen des Produkts in der aktuellen Zeile mit dem Produkt in der vorherigen Zeile kann diese Bedingung validiert werden.

Wenn die zweite Tabelle erstellt wird, wird die erste Tabelle tmp1Amounts mit einem „Drop Table“-Befehl gelöscht.

Abschließend erfolgt ein dritter Durchlauf der Daten, aber jetzt werden die Monate in umgekehrter Reihenfolge sortiert. So kann auch AmountMonthAfter berechnet werden.

„Order by“-Bedingungen geben an, wie die Tabelle sortiert wird. Ohne diese Bedingungen nutzt die Funktion Peek() eine beliebige Sortierung der internen Tabelle, was zu unvorhersehbaren Ergebnissen führen kann.

Ergebnis

Ergebnistabelle

Monat	Produkt	Amount	AmountMonthBefore	AmountMonthAfter
2022-01	A	8	-	6
2022-02	B	3	-	4
2022-03	A	6	8	6
2022-04	B	4	3	1
2022-05	A	6	6	5
2022-06	B	1	4	5
2022-01	A	5	6	7
2022-02	B	5	1	6
2022-03	A	7	5	4
2022-04	B	6	5	8
2022-05	A	4	7	-
2022-06	B	8	6	-

Beispiel 5

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Fügen Sie einem Arbeitsblatt in Ihrer App dann die Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um das Ergebnis anzuzeigen.

```
T1:
Load * inline [
Quarter, Value
2003q1, 10000
2003q1, 25000
2003q1, 30000
2003q2, 1250
2003q2, 55000
2003q2, 76200
2003q3, 9240
2003q3, 33150
```

```
2003q3, 89450
2003q4, 1000
2003q4, 3000
2003q4, 5000
2004q1, 1000
2004q1, 1250
2004q1, 3000
2004q2, 5000
2004q2, 9240
2004q2, 10000
2004q3, 25000
2004q3, 30000
2004q3, 33150
2004q4, 55000
2004q4, 76200
2004q4, 89450 ];

T2:
Load *, rangesum(SumVal,peek('AccSumVal')) as AccSumVal;
Load Quarter, sum(Value) as SumVal resident T1 group by Quarter;
```

Ergebnis

Ergebnistabelle

Quartal	SumVal	AccSumVal
2003q1	65000	65000
2003q2	132450	197450
2003q3	131840	329290
2003q4	9000	338290
2004q1	5250	343540
2004q2	24240	367780
2004q3	88150	455930
2004q4	220650	676580

Erläuterung

Die load-Anweisung **Load *, rangesum(SumVal,peek('AccSumVal')) as AccSumVal** umfasst einen rekursiven Aufruf, in dem vorherige Werte zum aktuellen Wert hinzugefügt werden. Dieser Vorgang wird verwendet, um eine Kumulierung von Werten im Skript zu berechnen.

Siehe auch:

Previous

Previous() liefert den Wert der **expr**-Formel, wobei für die Berechnung Daten aus dem letzten Datensatz verwendet werden, der nicht durch einen **where**-Zusatz ausgeschlossen wurde. Im ersten Datensatz einer internen Tabelle liefert diese Funktion NULL.

Syntax:

```
Previous(expr)
```

Rückgabe Datentyp: dual

Argumente:

Argumente

Argument	Beschreibung
expr	Die Formel oder das Feld mit den Daten, die gemessen werden sollen. Die Formel kann verschachtelte **previous()**-Funktionen enthalten, um weiter zurückliegende Datensätzen zu referenzieren. Die Daten werden direkt aus der Eingabequelle abgerufen, wodurch auch ein Bezug auf Felder möglich ist, die nicht in Qlik Sense geladen wurden, d. h. selbst wenn sie nicht in der assoziativen Datenbank gespeichert wurden.

Beschränkungen:

Im ersten Datensatz einer internen Tabelle liefert diese Funktion NULL.

Beispiel:

Geben Sie Folgendes in Ihr Ladeskript ein:

```
Sales2013:

Load *, (Sales - Previous(Sales) )as Increase Inline [

Month|Sales

1|12

2|13

3|15

4|17

5|21

6|21

7|22
```

```
8|23

9|32

10|35

11|40

12|41

] (delimiter is '|');
```

Durch den Einsatz der Funktion **Previous()** im **Load**-Befehl lässt sich der aktuelle Wert von Sales mit dem vorhergehenden Wert vergleichen und dieser in einem dritten Feld, Increase, verwenden.

Ergebnistabelle

Monat	Verkauf	Steigerung
1	12	-
2	13	1
3	15	2
4	17	2
5	21	4
6	21	0
7	22	1
8	23	1
9	32	9
10	35	3
11	40	5
12	41	1

Top - Diagrammfunktion

Top() interpretiert eine Formel in der ersten (obersten) Zeile eines Spaltenabschnitts in einer Tabelle. Die Zeile, für welche die Berechnung erfolgt, hängt vom Wert von **offset** ab; ist dieser vorhanden, wird standardmäßig die oberste Zeile verwendet. In Diagrammen erfolgt anders als in Tabellen mit **Top()** die Berechnung für die erste Zeile der aktuellen Spalte im entsprechenden Tabellendiagramm.

Syntax:

```
Top([TOTAL] expr [ , offset [,count ]])
```

Rückgabe Datentyp: dual

Argumente:

Argumente

Argument	Beschreibung
expr	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.
offset	Ist ein **offset** von n größer als 1 angegeben, wird die Formel anhand der Werte von n Zeilen berechnet, die unterhalb der obersten Zeile liegen. Bei einem negativen Startwert verhält sich die Funktion **Top** wie die Funktion **Bottom** mit dem entsprechenden positiven Startwert.
count	Ist ein dritter Parameter **count** größer als 1 angegeben, liefert die Funktion eine Menge von **count**-Werten, berechnet anhand der letzten **count**-Zeilen des Spaltenabschnitts. In diesem Formular kann die Funktion als Argument für eine der speziellen Abschnittsfunktionen dienen. *Bereichsfunktionen (page 1361)*
TOTAL	Wenn das Diagramm nur eine Dimension hat oder die Formel mit dem Zusatz **TOTAL** als Argument versehen ist, entspricht der Spaltenabschnitt der gesamten Spalte.

Ein Spaltenabschnitt ist als aufeinanderfolgende Teilmenge an Zellen definiert, welche dieselben Werte für die Dimensionen in der aktuellen Sortierreihenfolge besitzen. Inter-Record-Diagrammfunktionen werden im Spaltenabschnitt berechnet, wobei die rechts außen stehende Dimension im entsprechenden Tabellendiagramm ausgeschlossen wird. Befindet sich nur eine Dimension im Diagramm oder wird der Zusatz TOTAL angegeben, erfolgt die Berechnung über die gesamte Tabelle.

Hat das Diagramm dagegen mehrere vertikale Dimensionen, so umfasst der Spaltenabschnitt nur Zeilen, deren Werte in allen Dimensionen mit Ausnahme der in der Priorität der Sortierfolgen letzten Dimension übereinstimmen.

Beschränkungen:

- Die Rekursion liefert NULL.
- Das Sortieren nach y-Werten in Diagrammen oder nach Formelspalten in Tabellen ist nicht zulässig, wenn diese Diagrammfunktion in einer der Diagrammformeln verwendet wird. Diese Sortierungsoptionen werden daher automatisch deaktiviert. Wenn Sie diese Diagrammfunktion in einer Visualisierung oder Tabelle verwenden, wird die Sortierung der Visualisierung auf die sortierte Eingabe dieser Funktion zurückgesetzt.

Beispiele und Ergebnisse:

Beispiel: 1

Im Screenshot der in diesem Beispiel gezeigten Tabelle wird die Tabellenvisualisierung aus der Dimension **Customer** und den Kennzahlen `Sum(Sales)` und `Top(Sum(Sales))` erstellt.

Die Spalte **Top(Sum(Sales))** gibt 587 für alle Zeilen zurück, weil dies der Wert für die oberste Zeile ist: **Astrida**.

In der Tabelle werden auch komplexere Kennzahlen angezeigt: eine aus `Sum(Sales)+Top(Sum(Sales))` erstellte und eine mit **Top offset 3** beschriftete Kennzahl, die mithilfe der Formel `Sum(Sales)+Top(Sum(Sales), 3)` erstellt wird und über das Argument **offset** mit der Einstellung 3 verfügt. Sie fügt den Wert **Sum (Sales)** für die aktuelle Zeile zum Wert aus der dritten Zeile unter der obersten Zeile hinzu, d. h. die aktuelle Zeile plus den Wert für **Canutility**.

Beispiel 1

Top and Bottom

Customer	Sum(Sales)	Top(Sum(Sales))	Sum(Sales)+Top(Sum(Sales))	Top offset 3
Totals	2566	587	3153	3249
Astrida	587	587	1174	1270
Betacab	539	587	1126	1222
Canutility	683	587	1270	1366
Divadip	757	587	1344	1440

Beispiel: 2

In den Screenshots der in diesem Beispiel gezeigten Tabellen wurden den Visualisierungen weitere Dimensionen hinzugefügt: **Month** und **Product**. Bei Diagrammen mit mehr als einer Dimension hängt das Ergebnis der Formeln, welche die Funktionen **Above**, **Below**, **Top** und **Bottom** beinhalten, von der Reihenfolge ab, in der die Spaltendimensionen von Qlik Sense sortiert werden. Qlik Sense evaluiert die Funktionen auf Grundlage der Spaltenabschnitte, die sich aus der zuletzt sortierten Dimension ergeben haben. Die Sortierreihenfolge der Spalten kann im Eigenschaftsfenster unter **Sortierung** festgelegt werden und stimmt nicht zwangsläufig mit der Reihenfolge überein, in der die Spalten in der Tabelle angezeigt werden.

Erste Tabelle für Beispiel 2. Der Wert von Top für die Kennzahl First value basiert auf Month (Jan).

Customer	Product	Month	Sum(Sales)	First value
			2566	-
Astrida	AA	Jan	46	46
Astrida	AA	Feb	60	46
Astrida	AA	Mar	70	46
Astrida	AA	Apr	13	46
Astrida	AA	May	78	46
Astrida	AA	Jun	20	46
Astrida	AA	Jul	45	46
Astrida	AA	Aug	65	46
Astrida	AA	Sep	78	46
Astrida	AA	Oct	12	46
Astrida	AA	Nov	78	46
Astrida	AA	Dec	22	46

Zweite Tabelle für Beispiel 2. Der Wert von Top für die Kennzahl First value basiert auf Product (AA für Astrida).

Customer	Product	Month	Sum(Sales)	First value
			2566	-
Astrida	AA	Jan	46	46
Astrida	BB	Jan	46	46
Astrida	AA	Feb	60	60
Astrida	BB	Feb	60	60
Astrida	AA	Mar	70	70
Astrida	BB	Mar	70	70
Astrida	AA	Apr	13	13
Astrida	BB	Apr	13	13

Siehe Beispiel: 2 in der Funktion **Above** für weitere Details.

Beispiel 3

Beispiel: 3	**Ergebnis**
Die Funktion **Top** kann als Input für die Abschnittsfunktionen verwendet werden. Hier ein Beispiel: `RangeAvg (Top(Sum(Sales),1,3))`.	In den Argumenten für die Funktion **Top()** wird offset auf 1 und auf count gesetzt. Die Funktion ermittelt die Ergebnisse der Formel **Sum(Sales)** zu den drei Zeilen, beginnend bei der Zeile unterhalb der untersten Zeile im Spaltenabschnitt (weil offset=1), und den zwei Zeilen darunter (sofern Zeilen vorhanden sind). Diese drei Werte werden als Input für die Funktion RangeAvg() verwendet, die den Durchschnitt der Werte in der gelieferten Menge an Zahlen ermittelt. Eine Tabelle mit **Customer** als Dimension liefert folgende Ergebnisse für die Formel RangeAvg().
	Astrida 603 Betacab 603 Canutility 603 Divadip: 603

```
Monthnames:
LOAD *, Dual(MonthText,MonthNumber) as Month INLINE [
MonthText, MonthNumber
Jan, 1
Feb, 2
Mar, 3
Apr, 4
May, 5
Jun, 6
Jul, 7
Aug, 8
Sep, 9
Oct, 10
Nov, 11
Dec, 12
];

Sales2013:
Crosstable (MonthText, Sales) LOAD * inline [
Customer|Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec
Astrida|46|60|70|13|78|20|45|65|78|12|78|22
Betacab|65|56|22|79|12|56|45|24|32|78|55|15
Canutility|77|68|34|91|24|68|57|36|44|90|67|27
Divadip|57|36|44|90|67|27|57|68|47|90|80|94
] (delimiter is '|');
```

Siehe auch:

- *Bottom - Diagrammfunktion (page 1311)*
- *Above - Diagrammfunktion (page 1303)*
- *Sum - Diagrammfunktion (page 363)*
- *RangeAvg (page 1364)*
- *Bereichsfunktionen (page 1361)*

SecondaryDimensionality - Diagrammfunktion

SecondaryDimensionality() liefert die Zahl der Dimensionszeilen der Pivottabelle ohne aggregierten Inhalt, das heißt ohne Partialsummen und ausgeblendete Dimensionen. Diese Funktion entspricht der Funktion **dimensionality()** in horizontalen Pivottabellen.

Syntax:

```
SecondaryDimensionality( )
```

Rückgabe Datentyp: ganze Zahl

Beschränkungen:

- In anderen Diagrammtypen als Pivottabellen liefert die Funktion **SecondaryDimensionality** stets 0.
- Das Sortieren nach y-Werten in Diagrammen oder nach Formelspalten in Tabellen ist nicht zulässig, wenn diese Diagrammfunktion in einer der Diagrammformeln verwendet wird. Diese Sortierungsoptionen werden daher automatisch deaktiviert. Wenn Sie diese Diagrammfunktion in einer Visualisierung oder Tabelle verwenden, wird die Sortierung der Visualisierung auf die sortierte Eingabe dieser Funktion zurückgesetzt.

After - Diagrammfunktion

After() liefert das Ergebnis einer Formel, berechnet anhand der Dimensionswerte der nachfolgenden Formelspalte innerhalb desselben Zeilensegments der Pivottabelle.

Syntax:

```
after([TOTAL] expr [, offset [, count ]])
```

Das Sortieren nach y-Werten in Diagrammen oder nach Formelspalten in Tabellen ist nicht zulässig, wenn diese Diagrammfunktion in einer der Diagrammformeln verwendet wird. Diese Sortierungsoptionen werden daher automatisch deaktiviert. Wenn Sie diese Diagrammfunktion in einer Visualisierung oder Tabelle verwenden, wird die Sortierung der Visualisierung auf die sortierte Eingabe dieser Funktion zurückgesetzt.

Diese Funktion ist ausschließlich für Pivottabellen vorgesehen und liefert in allen anderen Diagrammtypen NULL.

Argumente:

Argument	Beschreibung
expr	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.
offset	Ist ein **offset** n größer als 1 angegeben, wird die Formel anhand der Werte von n Zeilen weiter rechts berechnet. Bei einem Startwert von "0" wird die Formel anhand der Werte der aktuellen Zeile berechnet. Bei einem negativen Startwert verhält sich die Funktion **After** wie die Funktion **Before** mit dem entsprechenden positiven Startwert.
count	Ist ein dritter Parameter **count** größer als 1 angegeben, liefert die Funktion einen Bereich von Werten – einen für jede Tabellenzeile bis zu einem Wert von **count**, ausgehend von den Zellen rechts der ursprünglichen Zelle.
TOTAL	Wenn das Diagramm nur eine Dimension hat oder die Formel mit dem Zusatz **TOTAL** als Argument versehen ist, entspricht der Spaltenabschnitt der gesamten Spalte.

In der letzten Spalte eines Zeilenabschnitts ist das Ergebnis NULL, da keine nachfolgende Spalte existiert.

Hat die Pivottabelle dagegen mehrere horizontale Dimensionen, so umfasst der Zeilenabschnitt nur Spalten, deren Werte in allen Dimensionen mit Ausnahme der in der Priorität der Sortierfolgen letzten Dimension übereinstimmen. Die Sortierfolge zwischen den Feldern für horizontale Dimensionen in Pivottabellen ergibt sich einfach durch die Reihenfolge der Dimensionen von oben nach unten.

Beispiel:

```
after( sum( Sales ))
after( sum( Sales ), 2 )
after( total sum( Sales ))
```

`rangeavg (after(sum(x),1,3))` liefert den Mittelwert der drei Ergebnisse der Funktion **sum(x)**, berechnet anhand der Werte der drei Spalten rechts neben der aktuellen Spalte.

Before - Diagrammfunktion

Before() liefert das Ergebnis einer Formel, berechnet anhand der Dimensionswerte der vorhergehenden Formelspalte innerhalb desselben Zeilenabschnitts der Pivottabelle.

Syntax:

```
before([TOTAL] expr [, offset [, count]])
```

Diese Funktion ist ausschließlich für Pivottabellen vorgesehen und liefert in allen anderen Diagrammtypen NULL.

Das Sortieren nach y-Werten in Diagrammen oder nach Formelspalten in Tabellen ist nicht zulässig, wenn diese Diagrammfunktion in einer der Diagrammformeln verwendet wird. Diese Sortierungsoptionen werden daher automatisch deaktiviert. Wenn Sie diese Diagrammfunktion in einer Visualisierung oder Tabelle verwenden, wird die Sortierung der Visualisierung auf die sortierte Eingabe dieser Funktion zurückgesetzt.

Argumente:

Argumente

Argument	Beschreibung
expr	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.
offset	Ist ein **offset** n größer als 1 angegeben, wird die Formel anhand der Werte von n Zeilen weiter links berechnet. Bei einem Startwert von "0" wird die Formel anhand der Werte der aktuellen Zeile berechnet. Bei einem negativen Startwert verhält sich die Funktion **Before** wie die Funktion **After** mit dem entsprechenden positiven Startwert.
count	Ist ein dritter Parameter **count** größer als 1 angegeben, liefert die Funktion einen Bereich von Werten – einen für jede Tabellenzeile bis zu einem Wert von **count**, ausgehend von den Zellen links der ursprünglichen Zelle.
TOTAL	Wenn das Diagramm nur eine Dimension hat oder die Formel mit dem Zusatz **TOTAL** als Argument versehen ist, entspricht der Spaltenabschnitt der gesamten Spalte.

In der ersten Spalte eines Zeilenabschnitts ist das Ergebnis NULL, da keine vorangehende Spalte existiert.

Hat die Pivottabelle dagegen mehrere horizontale Dimensionen, so umfasst der Zeilenabschnitt nur Spalten, deren Werte in allen Dimensionen mit Ausnahme der in der Priorität der Sortierfolgen letzten Dimension übereinstimmen.Die Sortierfolge zwischen den Feldern für horizontale Dimensionen in Pivottabellen ergibt sich einfach durch die Reihenfolge der Dimensionen von oben nach unten.

Beispiele:

```
before( sum( Sales ))
before( sum( Sales ), 2 )
before( total sum( Sales ))
```

`rangeavg (before(sum(x),1,3))` liefert den Mittelwert der drei Ergebnisse der Funktion **sum(x)**, berechnet anhand der Werte der drei Spalten links neben der aktuellen Spalte.

First - Diagrammfunktion

First() liefert das Ergebnis einer Formel, berechnet anhand der Dimensionswerte der ersten Formelspalte des Zeilenabschnitts der Pivottabelle. Diese Funktion ist ausschließlich für Pivottabellen vorgesehen und liefert in allen anderen Diagrammtypen NULL.

Das Sortieren nach y-Werten in Diagrammen oder nach Formelspalten in Tabellen ist nicht zulässig, wenn diese Diagrammfunktion in einer der Diagrammformeln verwendet wird. Diese Sortierungsoptionen werden daher automatisch deaktiviert. Wenn Sie diese Diagrammfunktion in einer Visualisierung oder Tabelle verwenden, wird die Sortierung der Visualisierung auf die sortierte Eingabe dieser Funktion zurückgesetzt.

Syntax:

```
first([TOTAL] expr [, offset [, count]])
```

Argumente:

Argumente

Argument	Beschreibung
expression	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.
offset	Ist ein **offset** n größer als 1 angegeben, wird die Formel anhand der Werte von n Zeilen weiter rechts berechnet. Bei einem Startwert von "0" wird die Formel anhand der Werte der aktuellen Zeile berechnet. Bei einem negativen Startwert verhält sich die Funktion **First** wie die Funktion **Last** mit dem entsprechenden positiven Startwert.
count	Ist ein dritter Parameter **count** größer als 1 angegeben, liefert die Funktion einen Bereich von Werten – einen für jede Tabellenzeile bis zu einem Wert von **count**, ausgehend von den Zellen rechts der ursprünglichen Zelle.
TOTAL	Wenn das Diagramm nur eine Dimension hat oder die Formel mit dem Zusatz **TOTAL** als Argument versehen ist, entspricht der Spaltenabschnitt der gesamten Spalte.

Hat die Pivottabelle dagegen mehrere horizontale Dimensionen, so umfasst der Zeilenabschnitt nur Spalten, deren Werte in allen Dimensionen mit Ausnahme der in der Priorität der Sortierfolgen letzten Dimension übereinstimmen.Die Sortierfolge zwischen den Feldern für horizontale Dimensionen in Pivottabellen ergibt sich einfach durch die Reihenfolge der Dimensionen von oben nach unten.

Beispiele:

```
first( sum( Sales ))
first( sum( Sales ), 2 )
first( total sum( Sales )
```

`rangeavg (first(sum(x),1,5))` liefert einen Mittelwert der Ergebnisse der Funktion **sum(x)**, berechnet anhand der Werte der ersten fünf Spalten des aktuellen Zeilensegments.

Last - Diagrammfunktion

Last() liefert das Ergebnis einer Formel, berechnet anhand der Dimensionswerte der letzten Formelspalte des Zeilenabschnitts der Pivottabelle. Diese Funktion ist ausschließlich für Pivottabellen vorgesehen und liefert in allen anderen Diagrammtypen NULL.

Das Sortieren nach y-Werten in Diagrammen oder nach Formelspalten in Tabellen ist nicht zulässig, wenn diese Diagrammfunktion in einer der Diagrammformeln verwendet wird. Diese Sortierungsoptionen werden daher automatisch deaktiviert. Wenn Sie diese Diagrammfunktion in einer Visualisierung oder Tabelle verwenden, wird die Sortierung der Visualisierung auf die sortierte Eingabe dieser Funktion zurückgesetzt.

Syntax:

```
last([TOTAL] expr [, offset [, count]])
```

Argumente:

Argumente

Argument	Beschreibung
expr	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.
offset	Ist ein **offset** n größer als 1 angegeben, wird die Formel anhand der Werte von n Zeilen weiter links berechnet. Bei einem Startwert von "0" wird die Formel anhand der Werte der aktuellen Zeile berechnet. Bei einem negativen Startwert verhält sich die Funktion **First** wie die Funktion **Last** mit dem entsprechenden positiven Startwert.
count	Ist ein dritter Parameter **count** größer als 1 angegeben, liefert die Funktion einen Bereich von Werten – einen für jede Tabellenzeile bis zu einem Wert von **count**, ausgehend von den Zellen links der ursprünglichen Zelle.
TOTAL	Wenn das Diagramm nur eine Dimension hat oder die Formel mit dem Zusatz **TOTAL** als Argument versehen ist, entspricht der Spaltenabschnitt der gesamten Spalte.

Hat die Pivottabelle dagegen mehrere horizontale Dimensionen, so umfasst der Zeilenabschnitt nur Spalten, deren Werte in allen Dimensionen mit Ausnahme der in der Priorität der Sortierfolgen letzten Dimension übereinstimmen.Die Sortierfolge zwischen den Feldern für horizontale Dimensionen in Pivottabellen ergibt sich einfach durch die Reihenfolge der Dimensionen von oben nach unten.

Beispiel:

```
last( sum( Sales ))
last( sum( Sales ), 2 )
last( total sum( Sales )
```

`rangeavg (last(sum(x),1,5))` liefert den Mittelwert der drei Ergebnisse der Funktion **sum(x)**, berechnet anhand der Werte der letzten fünf Spalten des Zeilenabschnitts.

ColumnNo - Diagrammfunktion

ColumnNo() liefert die Nummer der aktuellen Spalte innerhalb des Zeilenabschnitts der Pivottabelle. Die erste Spalte trägt die Nummer 1.

Syntax:

```
ColumnNo([total])
```

Argumente:

Argumente

Argument	Beschreibung
TOTAL	Wenn das Diagramm nur eine Dimension hat oder die Formel mit dem Zusatz **TOTAL** als Argument versehen ist, entspricht der Spaltenabschnitt der gesamten Spalte.

Hat die Pivottabelle dagegen mehrere horizontale Dimensionen, so umfasst der Zeilenabschnitt nur Spalten, deren Werte in allen Dimensionen mit Ausnahme der in der Priorität der Sortierfolgen letzten Dimension übereinstimmen.Die Sortierfolge zwischen den Feldern für horizontale Dimensionen in Pivottabellen ergibt sich einfach durch die Reihenfolge der Dimensionen von oben nach unten.

Das Sortieren nach y-Werten in Diagrammen oder nach Formelspalten in Tabellen ist nicht zulässig, wenn diese Diagrammfunktion in einer der Diagrammformeln verwendet wird. Diese Sortierungsoptionen werden daher automatisch deaktiviert. Wenn Sie diese Diagrammfunktion in einer Visualisierung oder Tabelle verwenden, wird die Sortierung der Visualisierung auf die sortierte Eingabe dieser Funktion zurückgesetzt.

Beispiel:

```
if( ColumnNo( )=1, 0, sum( Sales ) / before( sum( Sales )))
```

NoOfColumns - Diagrammfunktion

NoOfColumns() liefert die Anzahl der Spalten innerhalb des Zeilensegments in einer Pivottabelle.

Das Sortieren nach y-Werten in Diagrammen oder nach Formelspalten in Tabellen ist nicht zulässig, wenn diese Diagrammfunktion in einer der Diagrammformeln verwendet wird. Diese Sortierungsoptionen werden daher automatisch deaktiviert. Wenn Sie diese Diagrammfunktion in einer Visualisierung oder Tabelle verwenden, wird die Sortierung der Visualisierung auf die sortierte Eingabe dieser Funktion zurückgesetzt.

Syntax:

```
NoOfColumns([total])
```

Argumente:

Argumente

Argument	Beschreibung
TOTAL	Wenn das Diagramm nur eine Dimension hat oder die Formel mit dem Zusatz **TOTAL** als Argument versehen ist, entspricht der Spaltenabschnitt der gesamten Spalte.

Hat die Pivottabelle dagegen mehrere horizontale Dimensionen, so umfasst der Zeilenabschnitt nur Spalten, deren Werte in allen Dimensionen mit Ausnahme der in der Priorität der Sortierfolgen letzten Dimension übereinstimmen.Die Sortierfolge zwischen den Feldern für horizontale Dimensionen in Pivottabellen ergibt sich einfach durch die Reihenfolge der Dimensionen von oben nach unten.

Beispiel:

```
if( ColumnNo( )=NoOfColumns( ), 0, after( sum( Sales )))
```

8.17 Logische Funktionen

Dieser Abschnitt beschreibt Funktionen, mit denen logische Operationen durchgeführt werden. Alle Funktionen können sowohl im Datenladeskript als auch in den Diagrammformeln verwendet werden.

IsNum
Liefert -1 (True), wenn die Formel numerisch ist, anderenfalls 0 (False).

```
IsNum( expr )
```

IsText
Liefert -1 (True), wenn die Formel ein Textstring ist, anderenfalls 0 (False).

```
IsText( expr )
```

*Sowohl **IsNum** als auch **IsText** geben 0 zurück, wenn die Formel NULL ist.*

Beispiel:

Im folgenden Beispiel wird eine Inline-Tabelle mit gemischten Text- und numerischen Werten geladen und zwei Felder werden zum Überprüfen hinzugefügt, ob der Wert ein numerischer oder Textwert ist.

```
Load *, IsNum(Value), IsText(Value)
Inline [
Value
23
Green
Blue
12
33Red];
```

Die sich ergebende Tabelle sieht folgendermaßen aus:

Resulting table

Value	IsNum(Value)	IsText(Value)
23	-1	0
Green	0	-1
Blue	0	-1
12	-1	0
33Red	0	-1

8.18 Mapping-Funktionen

Dieser Abschnitt beschreibt Funktionen, mit denen Mapping-Tabellen gehandhabt werden. Eine Mapping-Tabelle kann während der Ausführung des Skripts zum Ersetzen von Feldwerten oder Feldnamen verwendet werden.

Mapping-Funktionen können nur im Datenladeskript verwendet werden.

Mapping-Funktionen – Übersicht

Jede Funktion wird nach der Übersicht genauer beschrieben. Sie können auch auf den Funktionsnamen in der Syntax klicken, um direkt auf die Details zu der spezifischen Funktion zuzugreifen.

ApplyMap
Die Skriptfunktion **ApplyMap** wird verwendet, um das Ergebnis einer Formel einer zuvor geladenen Mapping-Tabelle zuzuordnen.

```
ApplyMap ('mapname', expr [ , defaultexpr ] )
```

MapSubstring
Die Skriptfunktion **MapSubstring** wird zum Zuordnen einer Formel zu einer geladenen Mapping-Tabelle verwendet. Bei der Zuordnung wird die Groß-/Kleinschreibung beachtet, und es erfolgt nur ein Durchlauf von links nach rechts.

```
MapSubstring ('mapname', expr)
```

ApplyMap

Die Skriptfunktion **ApplyMap** wird verwendet, um das Ergebnis einer Formel einer zuvor geladenen Mapping-Tabelle zuzuordnen.

Syntax:

```
ApplyMap('map_name', expression [ , default_mapping ] )
```

Rückgabe Datentyp: dual

Argumente:

Argumente

Argument	Beschreibung
map_name	Der Name einer Mapping-Tabelle, die durch den Befehl **mapping load** oder **mapping select** erstellt wurde. Der Name muss in einfachen Anführungszeichen stehen. *Wenn Sie diese Funktion in einer makroerweiterten Variablen verwenden und auf eine nicht vorhandene Mapping-Tabelle verweisen, schlägt der Funktionsaufruf fehl, und das Feld wird nicht erstellt.*
expression	Die Formel, deren Ergebnis zugeordnet werden sollte.
default_ mapping	Sofern angegeben, wird dieser Wert als Standardwert verwendet, wenn die Mapping-Tabelle keine mit expression übereinstimmenden Werte enthält. Bei fehlender Angabe bleibt der Wert von expression unverändert.

Das Ausgabefeld von ApplyMap darf nicht den gleichen Namen wie eines der Eingabefelder haben. Dies könnte zu unerwarteten Ergebnissen führen. Nicht zu verwendendes Beispiel: `ApplyMap ('Map', A) as A`.

Beispiel:

In diesem Beispiel wird eine Liste von Verkäufern mit einem Ländercode geladen, der den Wohnsitzstaat enthält. Eine Tabelle, in der die Ländercodes den einzelnen Ländern zugeordnet sind, wird dazu verwendet, den Ländercode durch den tatsächlichen Namen des jeweiligen Landes zu ersetzen. Nur drei Länder sind in Mapping-Tabelle definiert, die anderen Ländercodes sind 'Rest of the world' zugeordnet.

```
// Load mapping table of country codes:
map1:
mapping LOAD *
Inline [
CCode, Country
Sw, Sweden
Dk, Denmark
No, Norway
] ;

// Load list of salesmen, mapping country code to country
// If the country code is not in the mapping table, put Rest of the world
Salespersons:
LOAD *,
ApplyMap('map1', CCode,'Rest of the world') As Country
Inline [
CCode, Salesperson
```

```
Sw, John
Sw, Mary
Sw, Per
Dk, Preben
Dk, Olle
No, Ole
Sf, Risttu
] ;

// We don't need the CCode anymore
Drop Field 'CCode';
```

Die sich ergebende Tabelle (Salespersons) sieht folgendermaßen aus:

Resulting table

Salesperson	**Country**
John	Sweden
Mary	Sweden
Per	Sweden
Preben	Denmark
Olle	Denmark
Ole	Norway
Risttu	Rest of the world

MapSubstring

Die Skriptfunktion **MapSubstring** wird zum Zuordnen einer Formel zu einer geladenen Mapping-Tabelle verwendet. Bei der Zuordnung wird die Groß-/Kleinschreibung beachtet, und es erfolgt nur ein Durchlauf von links nach rechts.

Syntax:

```
MapSubstring('map_name', expression)
```

Rückgabe Datentyp: String

Argumente:

Argumente

Argument	Beschreibung
map_name	Der Name einer Mapping-Tabelle, die durch die Befehle **mapping load** oder **mapping select** geladen wurde. Der Name der Mapping-Tabelle muss in einfachen Anführungszeichen stehen. *Wenn Sie diese Funktion in einer makroerweiterten Variablen verwenden und auf eine nicht vorhandene Mapping-Tabelle verweisen, schlägt der Funktionsaufruf fehl, und das Feld wird nicht erstellt.*
expression	Die Formel, deren Ergebnisse durch Teilstrings zugeordnet werden.

Beispiel:

In diesem Beispiel wird die Liste der Produktmodelle geladen. Jedes Modell verfügt über mehrere Attribute, die in einem zusammengesetzten Code beschrieben werden. Durch den Einsatz der Mapping-Tabelle mit MapSubstring können wir die Attributcodes auf eine Beschreibung ausweiten.

```
map2:
mapping LOAD *
Inline [
AttCode, Attribute
R, Red
Y, Yellow
B, Blue
C, Cotton
P, Polyester
S, Small
M, Medium
L, Large
] ;

Productmodels:
LOAD *,
MapSubString('map2', AttCode) as Description
Inline [
Model, AttCode
Twixie, R C S
Boomer, B P L
Raven, Y P M
Seedling, R C L
SeedlingPlus, R C L with hood
Younger, B C with patch
MultiStripe, R Y B C S/M/L
] ;
```

```
// We don't need the AttCode anymore
Drop Field 'AttCode';
```

Die sich ergebende Tabelle sieht folgendermaßen aus:

Resulting table

Model	Description
Twixie	Red Cotton Small
Boomer	Blue Polyester Large
Raven	Yellow Polyester Medium
Seedling	Red Cotton Large
SeedlingPlus	Red Cotton Large with hood
Younger	Blue Cotton with patch
MultiStripe	Red Yellow Blue Cotton Small/Medium/Large

8.19 Mathematische Funktionen

Dieser Abschnitt beschreibt Funktionen für mathematische Konstanten oder boolesche Werte. Diese Funktionen haben keine Parameter, doch die Klammern sind erforderlich.

Alle Funktionen können sowohl im Datenladeskript als auch in den Diagrammformeln verwendet werden.

e
Diese Funktion liefert die Basis des natürlichen Logarithmus, **e** (2,71828…).

```
e( )
```

false
Die Funktion liefert einen dualen Wert mit dem Textwert 'False' und dem numerischen Wert 0, der für logische Formeln verwendet werden kann.

```
false( )
```

pi
Die Funktion liefert den Wert von π (3,14159…).

```
pi( )
```

rand
Die Funktion liefert eine Zufallszahl zwischen 0 und 1. Diese kann zur Erstellung von Beispieldaten genutzt werden.

```
rand( )
```

Beispiel:

Dieses Beispielskript erstellt eine Tabelle mit 1000 Datensätzen mit zufällig ausgewählten Großbuchstaben, das heißt Zeichen im Bereich von 65 bis 91 (65+26).

```
Load
     Chr( Floor(rand() * 26) + 65) as UCaseChar,
     RecNo() as ID
     Autogenerate 1000;
```

true
Die Funktion liefert einen dualen Wert mit dem Textwert 'True' und dem numerischen Wert -1, der für logische Formeln verwendet werden kann.

```
true( )
```

8.20 NULL-Funktionen

Dieser Abschnitt beschreibt Funktionen für die Zurückgabe oder Erkennung von NULL-Werten.

Alle Funktionen können sowohl im Datenladeskript als auch in den Diagrammformeln verwendet werden.

NULL-Funktionen – Übersicht

Jede Funktion wird nach der Übersicht genauer beschrieben. Sie können auch auf den Funktionsnamen in der Syntax klicken, um direkt auf die Details zu der spezifischen Funktion zuzugreifen.

EmptyIsNull
Die Funktion **EmptyIsNull** konvertiert leere Zeichenfolgen in NULL. Daher gibt sie NULL zurück, wenn der Parameter eine leere Zeichenfolge ist. Andernfalls gibt sie den Parameter zurück.

```
EmptyIsNull (expr )
```

IsNull
Die Funktion **IsNull** überprüft, ob der Wert einer Formel NULL ist. Falls ja, wird -1 (True) ausgegeben, ansonsten 0 (False).

```
IsNull (expr )
```

Null
Die Funktion **Null** liefert einen NULL-Wert.

```
NULL( )
```

EmptyIsNull

Die Funktion **EmptyIsNull** konvertiert leere Zeichenfolgen in NULL. Daher gibt sie NULL zurück, wenn der Parameter eine leere Zeichenfolge ist. Andernfalls gibt sie den Parameter zurück.

Syntax:

```
EmptyIsNull(exp )
```

Beispiele und Ergebnisse:

Skriptbeispiele

Beispiel	Ergebnis
`EmptyIsNull(AdditionalComments)`	Diese Formel gibt alle leeren Zeichenfolgenwerte des Felds *AdditionalComments* als Null anstelle von leeren Zeichenfolgen zurück. Es werden nicht-leere Zeichenfolgen und Zahlen zurückgegeben.
`EmptyIsNull(PurgeChar (PhoneNumber, ' -()'))`	Diese Formel entfernt alle Bindestriche, Leerzeichen und Klammern aus dem Feld *PhoneNumber*. Wenn keine Zeichen übrig bleiben, gibt die Funktion „EmptyIsNull“ die leere Zeichenfolge als null zurück; eine leere Telefonnummer entspricht keiner Telefonnummer.

IsNull

Die Funktion **IsNull** überprüft, ob der Wert einer Formel NULL ist. Falls ja, wird -1 (True) ausgegeben, ansonsten 0 (False).

Syntax:

```
IsNull(expr )
```

*Ein String mit der Länge Null wird nicht als NULL betrachtet und **IsNull** wird als False ausgegeben.*

Beispiel: Datenladeskript

In diesem Beispiel wird eine Inline-Tabelle mit vier Zeilen geladen, in denen die ersten drei Zeilen entweder nichts - oder 'NULL' in der Spalte Value enthalten. Wir wandeln diese Werte in echte NULL-Wertrepräsentationen um, wobei der mittlere vorhergehende **LOAD**-Vorgang die **Null**-Funktion verwendet.

Der erste vorangehende **LOAD**-Vorgang fügt ein Feld hinzu, das überprüft, ob der Wert NULL ist. Dazu wird die **IsNull**-Funktion verwendet.

```
NullsDetectedAndConverted:

LOAD *,
If(IsNull(ValueNullConv), 'T', 'F') as IsItNull;

LOAD *,
If(len(trim(Value))= 0 or Value='NULL' or Value='-', Null(), Value ) as ValueNullConv;

LOAD * Inline
[ID, Value
0,
1,NULL
2,-
3,Value];
```

Daraus ergibt sich die folgende Tabelle. In der Spalte ValueNullConv werden die NULL-Werte durch - repräsentiert.

Resulting table

ID	Value	ValueNullConv	IsItNull
0		-	T
1	NULL	-	T
2	-	-	T
3	Value	Value	F

NULL

Die Funktion **Null** liefert einen NULL-Wert.

Syntax:

```
Null( )
```

Beispiel: Datenladeskript

In diesem Beispiel wird eine Inline-Tabelle mit vier Zeilen geladen, in denen die ersten drei Zeilen entweder nichts - oder 'NULL' in der Spalte Value enthalten. Diese Werte sollen in echte NULL-Wertrepräsentationen umgewandelt werden.

Der mittlere vorangehende **LOAD**-Befehl führt die Umwandlung mithilfe der **Null**-Funktion durch.

Der erste vorangehende **LOAD**-Vorgang fügt ein Feld hinzu, das überprüft, ob der Wert NULL ist. In diesem Beispiel dient dies nur zur Erläuterung.

```
NullsDetectedAndConverted:

LOAD *,
If(IsNull(ValueNullConv), 'T', 'F') as IsItNull;

LOAD *,
If(len(trim(Value))= 0 or Value='NULL' or Value='-', Null(), Value ) as ValueNullConv;

LOAD * Inline
[ID, Value
0,
1,NULL
2,-
3,Value];
```

Daraus ergibt sich die folgende Tabelle. In der Spalte ValueNullConv werden die NULL-Werte durch - repräsentiert.

Resulting table

ID	Value	ValueNullConv	IsItNull
0		-	T
1	NULL	-	T
2	-	-	T
3	Value	Value	F

8.21 Bereichsfunktionen

Die Mengenfunktionen sind Funktionen, die aus einer Vielzahl an Werten als Ergebnis einen einzelnen Wert berechnen. Alle Mengenfunktionen können sowohl im Datenladeskript als auch in den Diagrammformeln verwendet werden.

Zum Beispiel kann eine Mengenfunktion in einer Visualisierung einen einzelnen Wert aus einer Inter-Record-Vielzahl errechnen. Im Datenladeskript kann eine Mengenfunktion einen einzelnen Wert aus einer Vielzahl an Werten in einer internen Tabelle errechnen.

Bereichsfunktionen ersetzen die folgenden allgemeinen numerischen Funktionen: ***numsum****,* ***numavg****,* ***numcount****,* ***nummin*** *und* ***nummax****. Diese werden jetzt als veraltet angesehen.*

Einfache Abschnittsfunktionen

RangeMax

RangeMax() liefert den höchsten numerischen Wert in der Formel oder im Feld.

```
RangeMax (first_expr[, Expression])
```

RangeMaxString

RangeMaxString() liefert den letzten Wert in der Text-Sortierfolge, der in der Formel oder im Feld enthalten ist.

```
RangeMaxString (first_expr[, Expression])
```

RangeMin

RangeMin() liefert den niedrigsten numerischen Wert in der Formel oder im Feld.

```
RangeMin (first_expr[, Expression])
```

RangeMinString

RangeMinString() liefert den ersten Wert in der Text-Sortierfolge, der in der Formel oder im Feld enthalten ist.

```
RangeMinString (first_expr[, Expression])
```

RangeMode

RangeMode() liefert den am häufigsten vorkommenden Wert (Modalwert) in der Formel oder dem Feld.

```
RangeMode (first_expr[, Expression])
```

RangeOnly

RangeOnly() ist eine Dual-Funktion, die einen Wert liefert, wenn die Formel einen eindeutigen Wert ergibt. Ist das nicht der Fall, wird **NULL** ausgegeben.

```
RangeOnly (first_expr[, Expression])
```

RangeSum

RangeSum() gibt die Summe eines Wertebereichs zurück. Alle nicht numerischen Werte werden als 0 behandelt.

```
RangeSum (first_expr[, Expression])
```

Abschnittsfunktionen für Counter

RangeCount

RangeCount() liefert die Anzahl der Werte – numerisch und Text – in der Formel oder dem Feld.

```
RangeCount (first_expr[, Expression])
```

RangeMissingCount

RangeMissingCount() liefert die Anzahl der nicht numerischen Werte (einschließlich NULL) in der Formel oder dem Feld.

```
RangeMissingCount (first_expr[, Expression])
```

RangeNullCount

RangeNullCount() liefert die Anzahl der NULL-Werte in der Formel oder dem Feld.

```
RangeNullCount (first_expr[, Expression])
```

RangeNumericCount

RangeNumericCount() liefert die Anzahl der numerischen Werte in einer Formel oder einem Feld.

```
RangeNumericCount (first_expr[, Expression])
```

RangeTextCount

RangeTextCount() liefert die Anzahl der Text-Werte in einer Formel oder einem Feld.

```
RangeTextCount (first_expr[, Expression])
```

Statistische Abschnittsfunktionen

RangeAvg

RangeAvg() liefert den Durchschnittswert einer Reihe. Die Funktion arbeitet entweder mit eine Reihe von Werten oder einer Formel.

```
RangeAvg (first_expr[, Expression])
```

RangeCorrel

RangeCorrel() liefert den Korrelationskoeffizienten für zwei Datensätze. Der Korrelationskoeffizient ist eine Kennzahl, welche die Beziehung zwischen den Datensätzen kennzeichnet.

```
RangeCorrel (x_values , y_values[, Expression])
```

RangeFractile

RangeFractile() liefert den Wert, der dem n-ten **Fraktil** (Quantil) der Reihe an Zahlen entspricht.

```
RangeFractile (fractile, first_expr[ ,Expression])
```

RangeKurtosis

RangeKurtosis() liefert den Wert, welcher der Kurtosis der Zahlenreihe entspricht.

```
RangeKurtosis (first_expr[, Expression])
```

RangeSkew

RangeSkew() liefert den Wert, welcher der Schiefe der Reihe von Zahlen entspricht.

```
RangeSkew (first_expr[, Expression])
```

RangeStdev

RangeStdev() liefert die Standardabweichung einer Reihe von Zahlen.

```
RangeStdev (expr1[, Expression])
```

Finanz-Abschnittsfunktionen

RangeIRR

RangeIRR() liefert den internen Zinsfuß (Internal Rate of Return) für eine Reihe von Cashflows, die durch die Eingabewerte dargestellt sind.

```
RangeIRR (value[, value][, Expression])
```

RangeNPV

RangeNPV() liefert den momentanen Nettowert einer Investition, basierend auf einem Zinssatz und einer Reihe regelmäßig erfolgender Erträge (positive Werte) und Verluste (negative Werte). Das Ergebnis hat das Standardformat für Geldbeträge (siehe **money**).

```
RangeNPV (discount_rate, value[, value][, Expression])
```

RangeXIRR

RangeXIRR() gibt den internen Zinsfuß (Internal Rate of Return) (jährlich) für eine Reihe geplanter Geldflüsse zurück, die nicht unbedingt regelmäßig erfolgen müssen. Falls die Geldflüsse regelmäßig erfolgen, verwenden Sie die Funktion **RangeIRR** zur Berechnung des internen Zinsfußes.

```
RangeXIRR (values, dates[, Expression])
```

RangeXNPV

RangeXNPV() gibt den Nettobarwert (Net Present Value) einer Reihe geplanter (nicht unbedingt regelmäßig erfolgender) Geldflüsse zurück, repräsentiert durch die Wertepaare aus **pmt** und **date**. Alle Beträge werden auf ein 365-Tage-Jahr hochgerechnet.

```
RangeXNPV (discount_rate, values, dates[, Expression])
```

Siehe auch:

Inter-Record-Funktionen (page 1299)

RangeAvg

RangeAvg() liefert den Durchschnittswert einer Reihe. Die Funktion arbeitet entweder mit eine Reihe von Werten oder einer Formel.

Syntax:

```
RangeAvg(first_expr[, Expression])
```

Rückgabe Datentyp: numerisch

Argumente:

Die Argumente dieser Funktion können eine Inter-Record-Funktion enthalten, die auch wieder eine Liste der Werte liefert.

Argumente

Argument	Beschreibung
first_expr	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.
Expression	Optionale Formeln oder Felder, in denen die zu messenden Daten enthalten sind.

Beschränkungen:

Werden keine numerischen Werte gefunden, ist das Ergebnis NULL.

Beispiele und Ergebnisse:

Skriptbeispiele

Beispiele	Ergebnisse
`RangeAvg (1,2,4)`	Liefert 2,33333333
`RangeAvg (1,'xyz')`	Liefert 1
`RangeAvg (null( ), 'abc')`	Liefert NULL

Beispiel:

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Fügen Sie einem Arbeitsblatt in Ihrer App dann die Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um das Ergebnis anzuzeigen.

```
RangeTab3:
LOAD recno() as RangeID, RangeAvg(Field1,Field2,Field3) as MyRangeAvg INLINE [
Field1, Field2, Field3
10,5,6
2,3,7
8,2,8
18,11,9
5,5,9
9,4,2
];
```

Die entstehende Tabelle enthält die zurückgegebenen Werte von MyRangeAvg für jeden der Datensätze in der Tabelle.

Ergebnistabelle

RangeID	**MyRangeAvg**
1	7
2	4
3	6
4	12.666
5	6.333
6	5

Beispiel mit Formel:

```
RangeAvg (Above(MyField),0,3))
```

Liefert den Mittelwert der Ergebnisse einer Menge aus drei Werten der **MyField**-Funktion berechnet anhand der Werte der aktuellen Zeile und zwei Zeilen vor der aktuellen Zeile. Wird das dritte Argument als 3 angegeben, liefert die Funktion **Above()** drei Werte, sofern genügend Zeilen oberhalb vorhanden sind, die als Input für die Funktion **RangeAvg()** verwendet werden.

In Beispielen verwendete Daten:

*Deaktivieren Sie das Sortieren für **MyField**, damit Sie das gewünschte Ergebnis erhalten.*

Beispieldaten

MyField	RangeAvg (Above (MyField,0,3))	Comments
10	10	Weil dies die oberste Zeile ist, besteht die Menge nur aus einem Wert.
2	6	Es ist nur eine Zeile vor dieser Zeile vorhanden, daher ist die Menge: 10,2.
8	6.6666666667	Das Äquivalent zu RangeAvg(10,2,8)
18	9.333333333	-
5	10. 333333333	-
9	10.6666666667	-

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
] ;
```

Siehe auch:

- *Avg - Diagrammfunktion (page 419)*
- *Count - Diagrammfunktion (page 367)*

RangeCorrel

RangeCorrel() liefert den Korrelationskoeffizienten für zwei Datensätze. Der Korrelationskoeffizient ist eine Kennzahl, welche die Beziehung zwischen den Datensätzen kennzeichnet.

Syntax:

```
RangeCorrel(x_value , y_value[, Expression])
```

Rückgabe Datentyp: numerisch

Datenreihen sollten als (x,y)-Paare eingegeben werden. Wenn Sie beispielsweise zwei Datenserien (array 1 und array 2) berechnen möchten, wobei array 1 = 2,6,9 und array 2 = 3,8,4 ist, geben Sie `RangeCorrel (2,3,6,8,9,4)` ein. Das Ergebnis lautet dann 0,269.

Argumente:

Argumente

Argument	Beschreibung
x-value, y-value	Jeder Wert ist ein einzelner Wert oder eine Reihe von Werten, wie sie von einer Inter-Record-Funktion mit einem optionalen dritten Parameter ausgegeben wird. Zu jedem Wert oder jeder Reihe von Werten muss ein **x-value** oder eine Reihe von **y-values** existieren.
Expression	Optionale Formeln oder Felder, in denen die zu messenden Daten enthalten sind.

Beschränkungen:

Die Funktion benötigt mindestens zwei Koordinatenpaare, um ein Ergebnis berechnen zu können.

Textwerte, NULL-Werte und fehlende Werte geben NULL zurück.

Beispiele und Ergebnisse:

Funktionsbeispiele

Beispiele	Ergebnisse
`RangeCorrel (2,3,6,8,9,4,8,5)`	Liefert 0,2492. Diese Funktion kann in ein Skript geladen oder im Formel-Editor zu einer Visualisierung hinzugefügt werden.

Beispiel:

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Fügen Sie einem Arbeitsblatt in Ihrer App dann die Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um das Ergebnis anzuzeigen.

```
RangeList:
Load * Inline [
ID1|x1|y1|x2|y2|x3|y3|x4|y4|x5|y5|x6|y6
01|46|60|70|13|78|20|45|65|78|12|78|22
02|65|56|22|79|12|56|45|24|32|78|55|15
03|77|68|34|91|24|68|57|36|44|90|67|27
04|57|36|44|90|67|27|57|68|47|90|80|94
] (delimiter is '|');

XY:
LOAD recno() as RangeID, * Inline [
X|Y
2|3
6|8
9|4
8|5
](delimiter is '|');
```

In einer Tabelle mit ID1 als Dimension und der Kennzahl RangeCorrel(x1,y1,x2,y2,x3,y3,x4,y4,x5,y5,x6,y6)) liefert die Funktion **RangeCorrel()** den Wert von **Correl** über den Bereich von sechs x,y Paaren für jeden der ID1-Werte.

Ergebnistabelle

ID1	MyRangeCorrel
01	-0.9517
02	-0.5209
03	-0.5209
04	-0.1599

Beispiel:

```
XY:
LOAD recno() as RangeID, * Inline [
X|Y
2|3
6|8
9|4
8|5
](delimiter is '|');
```

In einer Tabelle mit RangeID als Dimension und der Kennzahl RangeCorrel(Below(X,0,4,BelowY,0,4)) verwendet die Funktion **RangeCorrel()** die Ergebnisse der **Below()**-Funktionen, die aufgrund des auf 4 festgelegten dritten Arguments (count) einen Bereich von vier x-y-Werten aus der geladenen Tabelle XY erzeugen.

Ergebnistabelle

RangeID	MyRangeCorrel2
01	0.2492
02	-0.9959
03	-1.0000
04	-

Der Wert für RangeID 01 ist derselbe wie bei der manuellen Eingabe von RangeCorrel(2,3,6,8,9,4,8,5). Für die übrigen Werte von RangeID sind die von der Below()-Funktion erzeugten Reihen: (6,8,9,4,8,5), (9,4,8,5) und (8,5), von denen die letzte ein NULL-Ergebnis erzeugt.

Siehe auch:

Correl - Diagrammfunktion (page 422)

RangeCount

RangeCount() liefert die Anzahl der Werte – numerisch und Text – in der Formel oder dem Feld.

Syntax:

```
RangeCount(first_expr[, Expression])
```

Rückgabe Datentyp: ganze Zahl

Argumente:

Die Argumente dieser Funktion können eine Inter-Record-Funktion enthalten, die auch wieder eine Liste der Werte liefert.

Argumente

Argument	Beschreibung
first_expr	Die Formel oder das Feld mit den Daten, die gezählt werden sollen.
Expression	Optionale Formeln oder Felder, die den Datenbereich enthalten, der gezählt werden soll.

Beschränkungen:

NULL-Werte werden nicht mitgezählt.

Beispiele und Ergebnisse:

Funktionsbeispiele

Beispiele	Ergebnisse
`RangeCount (1,2,4)`	Liefert 3
`RangeCount (2,'xyz')`	Liefert 2
`RangeCount (null( ))`	Liefert 0
`RangeCount (2,'xyz', null())`	Liefert 2

Beispiel:

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Fügen Sie einem Arbeitsblatt in Ihrer App dann die Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um das Ergebnis anzuzeigen.

```
RangeTab3:
LOAD recno() as RangeID, RangeCount(Field1,Field2,Field3) as MyRangeCount INLINE [
Field1, Field2, Field3
10,5,6
2,3,7
8,2,8
18,11,9
5,5,9
9,4,2
];
```

Die entstehende Tabelle enthält die zurückgegebenen Werte von MyRangeCount für jeden der Datensätze in der Tabelle.

Ergebnistabelle

RangeID	MyRangeCount
1	3
2	3
3	3
4	3
5	3
6	3

Beispiel mit Formel:

```
RangeCount (Above(MyField,1,3))
```

Liefert die Anzahl der in den drei Ergebnissen von **MyField** enthaltenen Werte. Die Angabe des ersten Arguments der Funktion **Above()** als 1 und des zweiten Arguments als 3 liefert die Werte der drei Felder oberhalb der aktuellen Zeile, sofern genügend Zeilen vorhanden sind, die als Input für die Funktion **RangeCount()** verwendet werden.

In Beispielen verwendete Daten:

Beispieldaten

MyField	RangeCount(Above(MyField,1,3))
10	0
2	1
8	2
18	3
5	3
9	3

In Beispielen verwendete Daten:

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
] ;
```

Siehe auch:

- *Count - Diagrammfunktion (page 367)*

RangeFractile

RangeFractile() liefert den Wert, der dem n-ten **Fraktil** (Quantil) der Reihe an Zahlen entspricht.

RangeFractile() verwendet beim Berechnen des Fraktils lineare Interpolation zwischen nächstgelegenen Rängen

Syntax:

```
RangeFractile(fractile, first_expr[, Expression])
```

Rückgabe Datentyp: numerisch

Argumente:

Die Argumente dieser Funktion können eine Inter-Record-Funktion enthalten, die auch wieder eine Liste der Werte liefert.

Argumente

Argument	Beschreibung
fractile	Eine Zahl zwischen 0 und 1, die dem zu berechnenden Fraktil (Quantil als Bruchteil ausgedrückt) entspricht.
first_expr	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.
Expression	Optionale Formeln oder Felder, in denen die zu messenden Daten enthalten sind.

Beispiele und Ergebnisse:

Funktionsbeispiele

Beispiele	Ergebnisse
`RangeFractile (0.24,1,2,4,6)`	Liefert 1,72
`RangeFractile(0.5,1,2,3,4,6)`	Liefert 3
`RangeFractile (0.5,1,2,5,6)`	Liefert 3,5

Beispiel:

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Fügen Sie einem Arbeitsblatt in Ihrer App dann die Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um das Ergebnis anzuzeigen.

```
RangeTab:
LOAD recno() as RangeID, RangeFractile(0.5,Field1,Field2,Field3) as MyRangeFrac INLINE [
Field1, Field2, Field3
10,5,6
2,3,7
8,2,8
18,11,9
5,5,9
```

```
9,4,2
];
```

Die entstehende Tabelle enthält die zurückgegebenen Werte von MyRangeFrac für jeden der Datensätze in der Tabelle.

Ergebnistabelle

RangeID	MyRangeFrac
1	6
2	3
3	8
4	11
5	5
6	4

Beispiel mit Formel:

```
RangeFractile (0.5, Above(Sum(MyField),0,3))
```

In diesem Beispiel enthält die Inter-Record-Funktion **Above()** den optionalen offset und count-Argumente. Dies bringt eine Reihe an Ergebnissen hervor, die als Input für die Mengenfunktionen verwendet werden können. In diesem Fall liefert `Above(Sum(MyField),0,3)` die Werte von `MyField` für die aktuelle Zeile und die zwei Zeilen davor. Diese Werte stellen den Input für die Funktion **RangeFractile()** bereit. Also entspricht dies für die unterste Zeile der Tabelle unten dem Wert von `RangeFractile(0.5, 3,4,6)`, das heißt, der Berechnung des 0.5-Fraktils für die Serien 3, 4 und 6. Für die ersten zwei Zeilen in der unten stehenden Tabelle wird die Anzahl der Werte in der Reihe entsprechend reduziert, wenn oberhalb der aktuellen Zeile keine Zeilen vorhanden sind. Ähnliche Ergebnisse werden für andere Inter-Record-Funktionen ausgegeben.

Beispieldaten

MyField	RangeFractile(0.5, Above(Sum(MyField),0,3))
1	1
2	1.5
3	2
4	3
5	4
6	5

In Beispielen verwendete Daten:

```
RangeTab:
LOAD * INLINE [
MyField
1
```

```
2
3
4
5
6
] ;
```

Siehe auch:

- *Above - Diagrammfunktion (page 1303)*
- *Fractile - Diagrammfunktion (page 426)*

RangeIRR

RangeIRR() liefert den internen Zinsfuß (Internal Rate of Return) für eine Reihe von Cashflows, die durch die Eingabewerte dargestellt sind.

Der interne Zinsfuß ist der Zinssatz einer Investition bei regelmäßig erfolgenden Auszahlungen (negative Werte) und Einzahlungen (positive Werte).

Diese Funktion verwendet eine vereinfachte Version der Newton-Methode zum Berechnen des internen Zinsfußes (IRR).

Syntax:

```
RangeIRR(value[, value][, Expression])
```

Rückgabe Datentyp: numerisch

Argumente

Argument	Beschreibung
value	Ein einzelner Wert oder eine Reihe von Werten, wie sie von einer Inter-Record-Funktion mit einem dritten Parameter ausgegeben wird. Die Funktion benötigt mindestens einen positiven und einen negativen Wert, um ein Ergebnis berechnen zu können.
Expression	Optionale Formeln oder Felder, in denen die zu messenden Daten enthalten sind.

Beschränkungen:

Textwerte, NULL-Werte und fehlende Werte werden ignoriert.

Beispieltabelle

Beispiele	Ergebnisse
`RangeIRR(-70000,12000,15000,18000,21000,26000)`	Liefert 0,0866

Beispiele	Ergebnisse

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Fügen Sie einem Arbeitsblatt in Ihrer App dann die Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um das Ergebnis anzuzeigen.
`RangeTab3:`
`LOAD *,`
`recno() as RangeID,`
`RangeIRR(Field1,Field2,Field3) as RangeIRR;`
`LOAD * INLINE [`
`Field1|Field2|Field3`
`-10000|5000|6000`
`-2000|NULL|7000`
`-8000|'abc'|8000`
`-1800|11000|9000`
`-5000|5000|9000`
`-9000|4000|2000`
`] (delimiter is '|');`
	Die entstehende Tabelle enthält die zurückgegebenen Werte von RangeIRR für jeden der Datensätze in der Tabelle.
	RangeID RangeIRR
	1 0.0639
	2 0.8708
	3 -
	4 5.8419
	5 0.9318
	6 -0.2566

Siehe auch:

- *Inter-Record-Funktionen (page 1299)*

RangeKurtosis

RangeKurtosis() liefert den Wert, welcher der Kurtosis der Zahlenreihe entspricht.

Syntax:

```
RangeKurtosis(first_expr[, Expression])
```

Rückgabe Datentyp: numerisch

Argumente:

Die Argumente dieser Funktion können eine Inter-Record-Funktion enthalten, die auch wieder eine Liste der Werte liefert.

Argumente

Argument	Beschreibung
first_expr	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.
Expression	Optionale Formeln oder Felder, in denen die zu messenden Daten enthalten sind.

Beschränkungen:

Werden keine numerischen Werte gefunden, ist das Ergebnis NULL.

Beispiele und Ergebnisse:

Funktionsbeispiele

Beispiele	Ergebnisse
`RangeKurtosis (1,2,4,7)`	Liefert -0,28571428571429

Siehe auch:

Kurtosis - Diagrammfunktion (page 434)

RangeMax

RangeMax() liefert den höchsten numerischen Wert in der Formel oder im Feld.

Syntax:

```
RangeMax(first_expr[, Expression])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
first_expr	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.
Expression	Optionale Formeln oder Felder, in denen die zu messenden Daten enthalten sind.

Beschränkungen:

Werden keine numerischen Werte gefunden, ist das Ergebnis NULL.

Beispiele und Ergebnisse:

Funktionsbeispiele

Beispiele	Ergebnisse
`RangeMax (1,2,4)`	Liefert 4
`RangeMax (1,'xyz')`	Liefert 1
`RangeMax (null( ), 'abc')`	Liefert NULL

Beispiel:

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Fügen Sie einem Arbeitsblatt in Ihrer App dann die Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um das Ergebnis anzuzeigen.

```
RangeTab3:
LOAD recno() as RangeID, RangeMax(Field1,Field2,Field3) as MyRangeMax INLINE [
Field1, Field2, Field3
10,5,6
2,3,7
8,2,8
18,11,9
5,5,9
9,4,2
];
```

Die entstehende Tabelle enthält die zurückgegebenen Werte von MyRangeMax für jeden der Datensätze in der Tabelle.

Ergebnistabelle

RangeID	MyRangeMax
1	10
2	7
3	8
4	18
5	9
6	9

Beispiel mit Formel:

```
RangeMax (Above(MyField,0,3))
```

Liefert den Maximalwert in einer Menge aus drei Werten der **MyField**-Funktion, berechnet anhand der Werte der aktuellen Zeile und zwei Zeilen vor der aktuellen Zeile. Wird das dritte Argument als 3 angegeben, liefert die Funktion **Above()** drei Werte, sofern genügend Zeilen oberhalb vorhanden sind, die als Input für die Funktion **RangeMax()** verwendet werden.

In Beispielen verwendete Daten:

*Deaktivieren Sie das Sortieren für **MyField**, damit Sie das gewünschte Ergebnis erhalten.*

Beispieldaten

MyField	RangeMax (Above(Sum(MyField),1,3))
10	10
2	10
8	10
18	18
5	18
9	18

In Beispielen verwendete Daten:

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
] ;
```

RangeMaxString

RangeMaxString() liefert den letzten Wert in der Text-Sortierfolge, der in der Formel oder im Feld enthalten ist.

Syntax:

```
RangeMaxString(first_expr[, Expression])
```

Rückgabe Datentyp: String

Argumente:

Die Argumente dieser Funktion können eine Inter-Record-Funktion enthalten, die auch wieder eine Liste der Werte liefert.

Argumente

Argument	Beschreibung
first_expr	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.
Expression	Optionale Formeln oder Felder, in denen die zu messenden Daten enthalten sind.

Beispiele und Ergebnisse:

Funktionsbeispiele

Beispiele	Ergebnisse
`RangeMaxString (1,2,4)`	Liefert 4
`RangeMaxString ('xyz','abc')`	Liefert 'xyz'
`RangeMaxString (5,'abc')`	Liefert 'abc'
`RangeMaxString (null( ))`	Liefert NULL

Beispiel mit Formel:

```
RangeMaxString (Above(MaxString(MyField),0,3))
```

Liefert das letzte (Text als Sortierfolge) der drei Ergebnisse der Funktion **MaxString(MyField)**, berechnet anhand der Werte der aktuellen Zeile und der beiden Zeilen über der aktuellen Zeile.

In Beispielen verwendete Daten:

*Deaktivieren Sie das Sortieren für **MyField**, damit Sie das gewünschte Ergebnis erhalten.*

Beispieldaten

MyField	RangeMaxString(Above(MaxString(MyField),0,3))
10	10
abc	abc
8	abc
def	def
xyz	xyz
9	xyz

In Beispielen verwendete Daten:

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
8
'def'
'xyz'
9
] ;
```

Siehe auch:

MaxString - Diagrammfunktion (page 558)

RangeMin

RangeMin() liefert den niedrigsten numerischen Wert in der Formel oder im Feld.

Syntax:

```
RangeMin(first_expr[, Expression])
```

Rückgabe Datentyp: numerisch

Argumente:

Argumente

Argument	Beschreibung
first_expr	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.
Expression	Optionale Formeln oder Felder, in denen die zu messenden Daten enthalten sind.

Beschränkungen:

Werden keine numerischen Werte gefunden, ist das Ergebnis NULL.

Beispiele und Ergebnisse:

Funktionsbeispiele

Beispiele	Ergebnisse
`RangeMin (1,2,4)`	Liefert 1
`RangeMin (1,'xyz')`	Liefert 1
`RangeMin (null( ), 'abc')`	Liefert NULL

Beispiel:

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Fügen Sie einem Arbeitsblatt in Ihrer App dann die Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um das Ergebnis anzuzeigen.

```
RangeTab3:
LOAD recno() as RangeID, RangeMin(Field1,Field2,Field3) as MyRangeMin INLINE [
Field1, Field2, Field3
10,5,6
2,3,7
8,2,8
18,11,9
5,5,9
9,4,2
];
```

Die entstehende Tabelle enthält die zurückgegebenen Werte von MyRangeMin für jeden der Datensätze in der Tabelle.

Ergebnistabelle

RangeID	MyRangeMin
1	5
2	2
3	2
4	9
5	5
6	2

Beispiel mit Formel:

```
RangeMin (Above(MyField,0,3)
```

Liefert den Minimalwert in einer Menge aus drei Werten der **MyField**-Funktion, berechnet anhand der Werte der aktuellen Zeile und zwei Zeilen vor der aktuellen Zeile. Wird das dritte Argument als 3 angegeben, liefert die Funktion **Above()** drei Werte, sofern genügend Zeilen oberhalb vorhanden sind, die als Input für die Funktion **RangeMin()** verwendet werden.

In Beispielen verwendete Daten:

Beispieldaten

MyField	RangeMin(Above(MyField,0,3))
10	10
2	2
8	2
18	2
5	5
9	5

In Beispielen verwendete Daten:

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
] ;
```

Siehe auch:

Min - Diagrammfunktion (page 354)

RangeMinString

RangeMinString() liefert den ersten Wert in der Text-Sortierfolge, der in der Formel oder im Feld enthalten ist.

Syntax:

```
RangeMinString(first_expr[, Expression])
```

Rückgabe Datentyp: String

Argumente:

Die Argumente dieser Funktion können eine Inter-Record-Funktion enthalten, die auch wieder eine Liste der Werte liefert.

Argumente

Argument	Beschreibung
first_expr	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.
Expression	Optionale Formeln oder Felder, in denen die zu messenden Daten enthalten sind.

Beispiele und Ergebnisse:

Funktionsbeispiele

Beispiele	Ergebnisse
`RangeMinString (1,2,4)`	Liefert 1
`RangeMinString ('xyz','abc')`	Liefert 'abc'
`RangeMinString (5,'abc')`	Liefert 5
`RangeMinString (null( ))`	Liefert NULL

Beispiel mit Formel:

```
RangeMinString (Above(MinString(MyField),0,3))
```

Liefert das erste (Text als Sortierfolge) der drei Ergebnisse der Funktion **MinString(MyField)**, berechnet anhand der Werte der aktuellen Zeile und der beiden Zeilen über der aktuellen Zeile.

In Beispielen verwendete Daten:

*Deaktivieren Sie das Sortieren für **MyField**, damit Sie das gewünschte Ergebnis erhalten.*

Beispieldaten

MyField	RangeMinString(Above(MinString(MyField),0,3))
10	10
abc	10
8	8
def	8
xyz	8
9	9

In Beispielen verwendete Daten:

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
8
'def'
'xyz'
9
] ;
```

Siehe auch:

MinString - Diagrammfunktion (page 561)

RangeMissingCount

RangeMissingCount() liefert die Anzahl der nicht numerischen Werte (einschließlich NULL) in der Formel oder dem Feld.

Syntax:

```
RangeMissingCount(first_expr[, Expression])
```

Rückgabe Datentyp: ganze Zahl

Argumente:

Die Argumente dieser Funktion können eine Inter-Record-Funktion enthalten, die auch wieder eine Liste der Werte liefert.

Argumente

Argument	Beschreibung
first_expr	Die Formel oder das Feld mit den Daten, die gezählt werden sollen.
Expression	Optionale Formeln oder Felder, die den Datenbereich enthalten, der gezählt werden soll.

Beispiele und Ergebnisse:

Funktionsbeispiele

Beispiele	Ergebnisse
`RangeMissingCount (1,2,4)`	Liefert 0
`RangeMissingCount (5,'abc')`	Liefert 1
`RangeMissingCount (null( ))`	Liefert 1

Beispiel mit Formel:

```
RangeMissingCount (Above(MinString(MyField),0,3))
```

Gibt die Anzahl der nicht numerischen Werte in den drei Ergebnissen der Funktion **MinString(MyField)** zurück, berechnet anhand der Werte der aktuellen Zeile und der beiden Zeilen über der aktuellen Zeile.

*Deaktivieren Sie das Sortieren für **MyField**, damit Sie das gewünschte Ergebnis erhalten.*

Beispieldaten

MyField	RangeMissingCount (Above(MinString (MyField),0,3))	Explanation
10	2	Liefert 2, da sich keine Zeilen über dieser Zeile befinden und daher 2 der 3 Werte fehlen.
abc	2	Gibt 2 zurück, da sich nur 1 Zeile über der aktuellen Zeile befindet und die aktuelle Zeile einen nicht numerischen Wert ('abc') aufweist.
8	1	Gibt 1 zurück, da 1 der 3 Zeilen einen nicht numerischen Wert ('abc') aufweist.
def	2	Gibt 2 zurück, da 2 der 3 Zeilen nicht numerische Werte ('def' und 'abc') aufweisen.
xyz	2	Gibt 2 zurück, da 2 der 3 Zeilen nicht numerische Werte (' xyz' und 'def') aufweisen.
9	2	Gibt 2 zurück, da 2 der 3 Zeilen nicht numerische Werte (' xyz' und 'def') aufweisen.

In Beispielen verwendete Daten:

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
8
```

```
'def'
'xyz'
9
] ;
```

Siehe auch:

- *MissingCount - Diagrammfunktion (page 371)*

RangeMode

RangeMode() liefert den am häufigsten vorkommenden Wert (Modalwert) in der Formel oder dem Feld.

Syntax:

```
RangeMode(first_expr {, Expression})
```

Rückgabe Datentyp: numerisch

Argumente:

Die Argumente dieser Funktion können eine Inter-Record-Funktion enthalten, die auch wieder eine Liste der Werte liefert.

Argumente

Argument	Beschreibung
first_expr	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.
Expression	Optionale Formeln oder Felder, in denen die zu messenden Daten enthalten sind.

Beschränkungen:

Wenn mehrere Werte am häufigsten vorkommen, ist das Ergebnis NULL.

Beispiele und Ergebnisse:

Funktionsbeispiele

Beispiele	Ergebnisse
`RangeMode (1,2,9,2,4)`	Liefert 2
`RangeMode ('a',4,'a',4)`	Liefert NULL
`RangeMode (null( ))`	Liefert NULL

Beispiel:

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Fügen Sie einem Arbeitsblatt in Ihrer App dann die Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um das Ergebnis anzuzeigen.

```
RangeTab3:
LOAD recno() as RangeID, RangeMode(Field1,Field2,Field3) as MyRangeMode INLINE [
```

```
Field1, Field2, Field3
10,5,6
2,3,7
8,2,8
18,11,9
5,5,9
9,4,2
];
```

Die entstehende Tabelle enthält die zurückgegebenen Werte von **MyRangeMode** für jeden der Datensätze in der Tabelle.

Ergebnistabelle

RangeID	**MyRangMode**
1	-
2	-
3	8
4	-
5	5
6	-

Beispiel mit Formel:

```
RangeMode (Above(MyField,0,3))
```

Liefert den am häufigsten vorkommenden Wert in den drei Ergebnissen der Funktion **MyField**, berechnet anhand der Werte der aktuellen Zeile und der beiden Zeilen über der aktuellen Zeile. Wird das dritte Argument als 3 angegeben, liefert die Funktion **Above()** drei Werte, sofern genügend Zeilen oberhalb vorhanden sind, die als Input für die Funktion **RangeMode()** verwendet werden.

Im Beispiel verwendete Daten:

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
] ;
```

*Deaktivieren Sie das Sortieren für **MyField**, damit Sie das gewünschte Ergebnis erhalten.*

Beispieldaten

MyField	RangeMode(Above(MyField,0,3))
10	Liefert 10, da keine darüberliegenden Zeilen vorhanden sind und der einzelne Wert am häufigsten vorkommt.
2	-
8	-
18	-
5	-
9	-

Siehe auch:

Mode - Diagrammfunktion (page 357)

RangeNPV

RangeNPV() liefert den momentanen Nettowert einer Investition, basierend auf einem Zinssatz und einer Reihe regelmäßig erfolgender Erträge (positive Werte) und Verluste (negative Werte). Das Ergebnis hat das Standardformat für Geldbeträge (siehe **money**).

Informationen zu Cashflows, die unregelmäßig erfolgen, finden Sie unter *RangeXNPV (page 1399)*.

Syntax:

```
RangeNPV(discount_rate, value[,value][, Expression])
```

Rückgabe Datentyp: numerisch

Argumente

Argument	Beschreibung
discount_ rate	Der Zinssatz pro Zeitraum.
value	Eine Ein- oder Auszahlung, die am Ende jedes Zeitraums erfolgt. Jeder Wert kann ein einzelner Wert oder eine Reihe von Werten sein, wie sie von einer Inter-Record-Funktion mit einem optionalen dritten Parameter ausgegeben wird.
Expression	Optionale Formeln oder Felder, in denen die zu messenden Daten enthalten sind.

Beschränkungen:

Textwerte, NULL-Werte und fehlende Werte werden ignoriert.

Beispiele	Ergebnisse
`RangeNPV(0.1,-10000,3000,4200,6800)`	Liefert 1188,44

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Fügen Sie einem Arbeitsblatt in Ihrer App dann die Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um das Ergebnis anzuzeigen.
`RangeTab3:`
`LOAD *,`
`recno() as RangeID,`
`RangeNPV(Field1,Field2,Field3) as RangeNPV;`
`LOAD * INLINE [`
`Field1|Field2|Field3`
`10|5|-6000`
`2|NULL|7000`
`8|'abc'|8000`
`18|11|9000`
`5|5|9000`
`9|4|2000`
`] (delimiter is '|');`
	Die entstehende Tabelle enthält die zurückgegebenen Werte von RangeNPV für jeden der Datensätze in der Tabelle.
	RangeID RangeNPV
	1 $-49.13
	2 $777.78
	3 $98.77
	4 $25.51
	5 $250.83
	6 $20.40

Siehe auch:

- *Inter-Record-Funktionen (page 1299)*

RangeNullCount

RangeNullCount() liefert die Anzahl der NULL-Werte in der Formel oder dem Feld.

Syntax:

```
RangeNullCount(first_expr [, Expression])
```

Rückgabe Datentyp: ganze Zahl

Argumente:

Die Argumente dieser Funktion können eine Inter-Record-Funktion enthalten, die auch wieder eine Liste der Werte liefert.

Argumente

Argument	Beschreibung
first_expr	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.
Expression	Optionale Formeln oder Felder, in denen die zu messenden Daten enthalten sind.

Beispiele und Ergebnisse:

Funktionsbeispiele

Beispiele	Ergebnisse
`RangeNullCount (1,2,4)`	Liefert 0
`RangeNullCount (5,'abc')`	Liefert 0
`RangeNullCount (null( ), null( ))`	Liefert 2

Beispiel mit Formel:

```
RangeNullCount (Above(Sum(MyField),0,3))
```

Liefert die Anzahl an NULL-Werten in den drei Ergebnissen der Funktion **Sum(MyField)**, berechnet anhand der Werte der aktuellen Zeile und der beiden Zeilen über der aktuellen Zeile.

*Kopieren von **MyField** im unteren Beispiel führt nicht zum Wert NULL.*

Beispieldaten

MyField	RangeNullCount(Above(Sum(MyField),0,3))
10	Liefert 2, da sich keine Zeilen über dieser Zeile befinden und daher 2 der 3 Werte fehlen (=NULL).
'abc'	Liefert 1, da sich nur eine Zeile über der aktuellen Zeile befindet und daher einer der drei Werte fehlt (=NULL).
8	Liefert 0, da keine der drei Zeilen ein NULL-Wert ist.

In Beispielen verwendete Daten:

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
8
] ;
```

Siehe auch:

NullCount - Diagrammfunktion (page 374)

RangeNumericCount

RangeNumericCount() liefert die Anzahl der numerischen Werte in einer Formel oder einem Feld.

Syntax:

```
RangeNumericCount(first_expr[, Expression])
```

Rückgabe Datentyp: ganze Zahl

Argumente:

Die Argumente dieser Funktion können eine Inter-Record-Funktion enthalten, die auch wieder eine Liste der Werte liefert.

Argumente

Argument	Beschreibung
first_expr	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.
Expression	Optionale Formeln oder Felder, in denen die zu messenden Daten enthalten sind.

Beispiele und Ergebnisse:

Funktionsbeispiele

Beispiele	Ergebnisse
`RangeNumericCount (1,2,4)`	Liefert 3
`RangeNumericCount (5,'abc')`	Liefert 1
`RangeNumericCount (null( ))`	Liefert 0

Beispiel mit Formel:

```
RangeNumericCount (Above(MaxString(MyField),0,3))
```

Liefert die Anzahl von numerischen Werten in den drei Ergebnissen der Funktion **MaxString(MyField)**, berechnet anhand der Werte der aktuellen Zeile und der beiden Zeilen über der aktuellen Zeile.

*Deaktivieren Sie das Sortieren für **MyField**, damit Sie das gewünschte Ergebnis erhalten.*

Beispieldaten

MyField	RangeNumericCount(Above(MaxString(MyField),0,3))
10	1
abc	1
8	2

MyField	RangeNumericCount(Above(MaxString(MyField),0,3))
def	1
xyz	1
9	1

In Beispielen verwendete Daten:

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
8
def
xyz
9
] ;
```

Siehe auch:

NumericCount - Diagrammfunktion (page 377)

RangeOnly

RangeOnly() ist eine Dual-Funktion, die einen Wert liefert, wenn die Formel einen eindeutigen Wert ergibt. Ist das nicht der Fall, wird **NULL** ausgegeben.

Syntax:

```
RangeOnly(first_expr[, Expression])
```

Rückgabe Datentyp: dual

Argumente:

Die Argumente dieser Funktion können eine Inter-Record-Funktion enthalten, die auch wieder eine Liste der Werte liefert.

Argument	Beschreibung
first_expr	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.
Expression	Optionale Formeln oder Felder, in denen die zu messenden Daten enthalten sind.

Beispiele und Ergebnisse:

Beispiele	Ergebnisse
`RangeOnly (1,2,4)`	Liefert NULL

Beispiele	Ergebnisse
`RangeOnly (5,'abc')`	Liefert NULL
`RangeOnly (null( ), 'abc')`	Liefert 'abc'
`RangeOnly(10,10,10)`	Liefert 10

Siehe auch:

Only - Diagrammfunktion (page 360)

RangeSkew

RangeSkew() liefert den Wert, welcher der Schiefe der Reihe von Zahlen entspricht.

Syntax:

```
RangeSkew(first_expr[, Expression])
```

Rückgabe Datentyp: numerisch

Argumente:

Die Argumente dieser Funktion können eine Inter-Record-Funktion enthalten, die auch wieder eine Liste der Werte liefert.

Argumente

Argument	Beschreibung
first_expr	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.
Expression	Optionale Formeln oder Felder, in denen die zu messenden Daten enthalten sind.

Beschränkungen:

Werden keine numerischen Werte gefunden, ist das Ergebnis NULL.

Beispiele und Ergebnisse:

Funktionsbeispiele

Beispiele	Ergebnisse
`rangeskew (1,2,4)`	Liefert 0,93521952958283
`rangeskew (above (SalesValue,0,3))`	Liefert die Schiefe des Berreichs der drei Werte, die von der Funktion above() geliefert werden, berechnet anhand der Werte der aktuellen Zeile und der beiden Zeilen über der aktuellen Zeile.

Im Beispiel verwendete Daten:

Beispieldaten

CustID	RangeSkew(Above(SalesValue,0,3))
1-20	-, -, 0.5676, 0.8455, 1.0127, -0.8741, 1.7243, -1.7186, 1.5518, 1.4332, 0, 1.1066, 1.3458, 1.5636, 1.5439, 0.6952, -0.3766

```
SalesTable:
LOAD recno() as CustID, * inline [
SalesValue
101
163
126
139
167
86
83
22
32
70
108
124
176
113
95
32
42
92
61
21
] ;
```

Siehe auch:

Skew - Diagrammfunktion (page 467)

RangeStdev

RangeStdev() liefert die Standardabweichung einer Reihe von Zahlen.

Syntax:

```
RangeStdev(first_expr[, Expression])
```

Rückgabe Datentyp: numerisch

Argumente:

Die Argumente dieser Funktion können eine Inter-Record-Funktion enthalten, die auch wieder eine Liste der Werte liefert.

Argumente

Argument	Beschreibung
first_expr	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.
Expression	Optionale Formeln oder Felder, in denen die zu messenden Daten enthalten sind.

Beschränkungen:

Werden keine numerischen Werte gefunden, ist das Ergebnis NULL.

Beispiele und Ergebnisse:

Funktionsbeispiele

Beispiele	Ergebnisse
`RangeStdev (1,2,4)`	Liefert 1,5275252316519
`RangeStdev (null( ))`	Liefert NULL
`RangeStdev (above (SalesValue),0,3))`	Liefert die Schiefe des Bereichs der drei Werte, die von der Funktion above() geliefert werden, berechnet anhand der Werte der aktuellen Zeile und der beiden Zeilen über der aktuellen Zeile.

Im Beispiel verwendete Daten:

Beispieldaten

CustID	RangeStdev(SalesValue, 0,3))
1-20	-,43.841, 34.192, 18.771, 20.953, 41.138, 47.655, 36.116, 32.716, 25.325, 38,000, 27.737, 35.553, 33.650, 42.532, 33.858, 32.146, 25.239, 35.595

```
SalesTable:
LOAD recno() as CustID, * inline [
SalesValue
101
163
126
139
167
86
83
22
32
70
108
124
176
113
95
32
42
92
61
```

```
21
] ;
```

Siehe auch:

- *Stdev - Diagrammfunktion (page 469)*

RangeSum

RangeSum() gibt die Summe eines Wertebereichs zurück. Alle nicht numerischen Werte werden als 0 behandelt.

Syntax:

```
RangeSum(first_expr[, Expression])
```

Rückgabe Datentyp: numerisch

Argumente:

Die Argumente dieser Funktion können eine Inter-Record-Funktion enthalten, die auch wieder eine Liste der Werte liefert.

Argumente

Argument	Beschreibung
first_expr	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.
Expression	Optionale Formeln oder Felder, in denen die zu messenden Daten enthalten sind.

Beschränkungen:

Die Funktion **RangeSum** behandelt alle nicht numerischen Werte als 0.

Beispiele und Ergebnisse:

Beispiele

Beispiele	Ergebnisse
`RangeSum (1,2,4)`	Liefert 7
`RangeSum (5,'abc')`	Liefert 5
`RangeSum (null( ))`	Liefert 0

Beispiel:

Fügen Sie Ihrer App ein Beispielskript hinzu und führen Sie dieses aus. Fügen Sie einem Arbeitsblatt in Ihrer App dann die Felder hinzu, die in der Ergebnisspalte aufgeführt sind, um das Ergebnis anzuzeigen.

```
RangeTab3:

LOAD recno() as RangeID, Rangesum(Field1,Field2,Field3) as MyRangeSum INLINE [
```

```
Field1, Field2, Field3

10,5,6

2,3,7

8,2,8

18,11,9

5,5,9

9,4,2
];
```

Die entstehende Tabelle enthält die zurückgegebenen Werte von MyRangeSum für jeden der Datensätze in der Tabelle.

Ergebnistabelle

RangeID	MyRangeSum
1	21
2	12
3	18
4	38
5	19
6	15

Beispiel mit Formel:

```
RangeSum (Above(MyField,0,3))
```

Liefert die Summe der drei Werte der Funktion **MyField)**: anhand der Werte der aktuellen Zeile und der beiden Zeilen über der aktuellen Zeile. Wird das dritte Argument als 3 angegeben, liefert die Funktion **Above()** drei Werte, sofern genügend Zeilen oberhalb vorhanden sind, die als Input für die Funktion **RangeSum()** verwendet werden.

In Beispielen verwendete Daten:

*Deaktivieren Sie das Sortieren für **MyField**, damit Sie das gewünschte Ergebnis erhalten.*

Beispieldaten

MyField	RangeSum(Above(MyField,0,3))
10	10
2	12

MyField	RangeSum(Above(MyField,0,3))
8	20
18	28
5	31
9	32

In Beispielen verwendete Daten:

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
] ;
```

Siehe auch:

- *Sum - Diagrammfunktion (page 363)*
- *Above - Diagrammfunktion (page 1303)*

RangeTextCount

RangeTextCount() liefert die Anzahl der Text-Werte in einer Formel oder einem Feld.

Syntax:

```
RangeTextCount(first_expr[, Expression])
```

Rückgabe Datentyp: ganze Zahl

Argumente:

Die Argumente dieser Funktion können eine Inter-Record-Funktion enthalten, die auch wieder eine Liste der Werte liefert.

Argument

Argument	Beschreibung
first_expr	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.
Expression	Optionale Formeln oder Felder, in denen die zu messenden Daten enthalten sind.

Beispiele und Ergebnisse:

Funktionsbeispiele

Beispiele	Ergebnisse
`RangeTextCount (1,2,4)`	Liefert 0
`RangeTextCount (5,'abc')`	Liefert 1
`RangeTextCount (null( ))`	Liefert 0

Beispiel mit Formel:

```
RangeTextCount (Above(MaxString(MyField),0,3))
```

Liefert die Anzahl von Textwerten in den drei Ergebnissen der Funktion **MaxString(MyField)**, berechnet anhand der Werte der aktuellen Zeile und der beiden Zeilen über der aktuellen Zeile.

In Beispielen verwendete Daten:

*Deaktivieren Sie das Sortieren für **MyField**, damit Sie das gewünschte Ergebnis erhalten.*

Beispieldaten

MyField	MaxString(MyField)	RangeTextCount(Above(Sum(MyField),0,3))
10	10	0
abc	abc	1
8	8	1
def	def	2
xyz	xyz	2
9	9	2

In Beispielen verwendete Daten:

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
8
null()
'xyz'
9
] ;
```

Siehe auch:

TextCount - Diagrammfunktion (page 381)

RangeXIRR

RangeXIRR() gibt den internen Zinsfuß (Internal Rate of Return) (jährlich) für eine Reihe geplanter Geldflüsse zurück, die nicht unbedingt regelmäßig erfolgen müssen. Falls die Geldflüsse regelmäßig erfolgen, verwenden Sie die Funktion **RangeIRR** zur Berechnung des internen Zinsfußes.

Die XIRR-Funktionalität von Qlik (die Funktionen **XIRR()** und **RangeXIRR()**) verwendet die folgende Gleichung, die nach dem Wert `Rate` aufgelöst wird, um den korrekten XIRR-Wert zu bestimmen:

```
XNPV(Rate, pmt, date) = 0
```

Die Gleichung wird anhand einer vereinfachten Version der Newton-Methode aufgelöst.

Syntax:

```
RangeXIRR(value, date{, value, date})
```

Rückgabe Datentyp: numerisch

Argumente

Argument	Beschreibung
value	Ein Geldfluss oder eine Reihe von Geldflüssen, die geplanten Zahlungszeitpunkten entsprechen. Die Funktion benötigt mindestens einen positiven und einen negativen Wert, um ein Ergebnis berechnen zu können.
date	Ein Zahlungszeitpunkt oder eine Reihe von Zahlungszeitpunkten für die Geldflusszahlungen.

Bei der Arbeit mit dieser Funktion gelten die folgenden Einschränkungen:

- Textwerte, NULL-Werte und fehlende Werte werden ignoriert.
- Alle Beträge werden auf ein 365-Tage-Jahr hochgerechnet.
- Diese Funktion erfordert mindestens eine gültige negative und einen gültige positive Zahlung (mit entsprechendem gültigem Datum). Wenn diese Zahlungen nicht angegeben werden, wird ein Wert von NULL zurückgegeben.

Die folgenden Themen können Sie bei der Arbeit mit dieser Funktion unterstützen:

- *RangeXNPV (page 1399)*: Verwenden Sie diese Funktion zum Berechnen des Nettobarwerts einer Reihe von Geldflüssen, die nicht unbedingt regelmäßig erfolgen.
- *XIRR (page 395)*: Die Funktion **XIRR()** berechnet den aggregierten internen Zinsfuß (Internal Rate of Return) (jährlich) für eine Reihe von Geldflüssen. Diese müssen nicht unbedingt regelmäßig erfolgen.

In den einzelnen Versionen von Qlik Sense Client-Managed bestehen Unterschiede bei dem zugrunde liegenden Algorithmus, der von dieser Funktion verwendet wird. Informationen zu kürzlichen Aktualisierungen am Algorithmus finden Sie im Support-Artikel über XIRR-Funktion – Behebung und Aktualisierung.

Beispiele und Ergebnisse:

Beispiele und Ergebnisse

Beispiele	Ergebnisse
`RangeXIRR(-2500,'2008-01-01',2750,'2008-09-01')`	Liefert 0,1532

Siehe auch:

- *RangeIRR (page 1373)*
- *RangeXNPV (page 1399)*
- *XIRR (page 395)*
- XIRR-Funktion – Behebung und Aktualisierung

RangeXNPV

RangeXNPV() gibt den Nettobarwert (Net Present Value) einer Reihe geplanter (nicht unbedingt regelmäßig erfolgender) Geldflüsse zurück, repräsentiert durch die Wertepaare aus **pmt** und **date**. Alle Beträge werden auf ein 365-Tage-Jahr hochgerechnet.

Syntax:

```
RangeXNPV(discount_rate, value, date{, value, date})
```

Rückgabe Datentyp: numerisch

Argumente

Argument	Beschreibung
discount_rate	**discount_rate** ist die jährliche Rate, um die die Zahlungen diskontiert werden sollen.
value	Ein Geldfluss oder eine Reihe von Geldflüssen, die geplanten Zahlungszeitpunkten entsprechen. Jeder Wert kann ein einzelner Wert oder eine Reihe von Werten sein, wie sie von einer Inter-Record-Funktion mit einem optionalen dritten Parameter ausgegeben wird. Die Funktion benötigt mindestens einen positiven und einen negativen Wert, um ein Ergebnis berechnen zu können.
date	Ein Zahlungszeitpunkt oder eine Reihe von Zahlungszeitpunkten für die Geldflusszahlungen.

Bei der Arbeit mit dieser Funktion gelten die folgenden Einschränkungen:

- Textwerte, NULL-Werte und fehlende Werte werden ignoriert.
- Alle Beträge werden auf ein 365-Tage-Jahr hochgerechnet.

Beispiel – Skript

Ladeskript und Ergebnisse

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Finanzdaten in einer Tabelle namens `RangeTab3`.
- Verwendung der Funktion **RangeXNPV()** zum Berechnen des Nettobarwerts.

Ladeskript

```
RangeTab3:
LOAD *,
recno() as RangeID,
RangeXNPV(DiscountRate,Value1,Date1,Value2,Date2) as RangeXNPV;
LOAD * INLINE [
DiscountRate|Value1|Date1|Value2|Date2
0.1|-100|2021-01-01|100|2022-01-01|
0.1|-100|2021-01-01|110|2022-01-01|
0.1|-100|2021-01-01|125|2022-01-01|
] (delimiter is '|');
```

Ergebnisse

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgenden Felder als Dimensionen hinzu:

- `RangeID`
- `RangeXNPV`

Ergebnistabelle

RangeID	RangeXNPV
1	-$9.09
2	-$0.00
3	$13.64

Beispiel – Diagrammformel

Ladeskript und Diagrammformel

Übersicht

Öffnen Sie den Dateneditor und fügen Sie das Ladeskript unten in eine neue Registerkarte ein.

Das Ladeskript umfasst:

- Finanzdaten in einer Tabelle namens `RangeTab3`.
- Verwendung der Funktion **RangeXNPV()** zum Berechnen des Nettobarwerts.

Ladeskript

```
RangeTab3:
LOAD *,
recno() as RangeID,
RangeXNPV(DiscountRate,Value1,Date1,Value2,Date2) as RangeXNPV;
LOAD * INLINE [
DiscountRate|Value1|Date1|Value2|Date2
0.1|-100|2021-01-01|100|2022-01-01|
0.1|-100|2021-01-01|110|2022-01-01|
0.1|-100|2021-01-01|125|2022-01-01|
] (delimiter is '|');
```

Ergebnisse

Gehen Sie folgendermaßen vor:

Laden Sie die Daten und öffnen Sie ein Arbeitsblatt. Erstellen Sie eine neue Tabelle und fügen Sie die folgende Berechnung als Kennzahl hinzu.

```
=RangeXNPV(0.1, -2500,'2008-01-01',2750,'2008-09-01')
```

Ergebnistabelle

=XIRR(Payments, Date)
$80.25

Siehe auch:

XNPV (page 402)

8.22 Relationale Funktionen

Dies ist eine Gruppe Funktionen, die Eigenschaften einzelner Dimensionswerte in einem Diagramm berechnen und bereits aggregierte Zahlen verwenden.

Die Funktionen sind relational in dem Sinn, dass die Funktionsausgabe nicht nur von dem Wert des Datenpunkts selbst abhängt, sondern auch von der Beziehung des Werts zu anderen Datenpunkten. Beispielsweise kann eine Rangfolge nicht ohne Vergleich mit anderen Dimensionswerten berechnet werden.

Diese Funktionen können ausschließlich in den Formeln von Diagrammen verwendet werden. Sie können nicht im Ladeskript verwendet werden.

Eine Dimension wird im Diagramm benötigt, da sie die anderen Datenpunkte definiert, die für den Vergleich erforderlich sind. Daher ist eine relationale Funktion in einem Diagramm ohne Dimensionen nicht sinnvoll (z. B. einem KPI-Objekt).

Rangfolgefunktionen

Ferner können beim Gebrauch von Inter-Record-Funktionen keine Nullwerte weggelassen werden. NULL-Werte werden ignoriert.

Rank

Rank() berechnet die Zeilen des Diagramms in der Formel und zeigt für jede Zeile die relative Position des Wertes der Dimension an, die in der Formel berechnet wird. Beim Auswerten der Formel vergleicht die Funktion ihr Ergebnis mit den Ergebnissen für andere Zeilen innerhalb des Spaltenabschnitts und liefert den Rang der Zeile innerhalb des Spaltenabschnitts.

```
Rank - Diagrammfunktion([TOTAL [<fld {, fld}>]] expr[, mode[, fmt]])
```

HRank

HRank() wertet die Formel aus und vergleicht ihr Ergebnis mit den Ergebnissen für andere Spalten desselben Zeilenabschnitts der Pivottabelle. Die Funktion liefert anschließend die Rangfolge der Spalte innerhalb des Abschnitts.

```
HRank - Diagrammfunktion([TOTAL] expr[, mode[, fmt]])
```

Clustering-Funktionen

KMeans2D

Die Eigenschaftengruppe **Site-Lizenz** enthält Eigenschaften im Zusammenhang mit der Lizenz für das Qlik Sense System. Alle Felder sind Pflichtfelder und dürfen nicht leer sein.

Site-Lizenzeigenschaften

Eigenschaftsname	Beschreibung
Benutzername	Der Benutzername des Qlik Sense Produktbesitzers
Benutzerorganisation	Der Name der Organisation, bei der der Qlik Sense Produktbesitzer Mitglied ist
Seriennummer	Die Seriennummer, die der Qlik Sense Software zugewiesen ist
Kontrollnummer	Die Kontrollnummer, die der Qlik Sense Software zugewiesen ist
LEF-Zugriff	Die License Enabler File (LEF), die der Qlik Sense Software zugewiesen ist.

KMeans2D() wertet die Zeilen des Diagramms aus, indem K-means-Clustering angewandt wird. Für jede Diagrammzeile wird die Cluster-ID des Clusters angezeigt, dem dieser Datenpunkt zugewiesen wurde. Die vom Clustering-Algorithmus verwendeten Spalten werden von den Parametern coordinate_1 bzw. coordinate_2 festgelegt. Es handelt sich bei beiden um Aggregierungen. Die Anzahl der erstellten Cluster wird durch den Parameter num_clusters bestimmt. Daten können optional mit dem Normparameter normalisiert werden.

```
KMeans2D - Diagrammfunktion(num_clusters, coordinate_1, coordinate_2 [,
norm])
```

KMeansND

KMeansND() wertet die Zeilen des Diagramms aus, indem K-means-Clustering angewandt wird. Für jede Diagrammzeile wird die Cluster-ID des Clusters angezeigt, dem dieser Datenpunkt zugewiesen wurde. Die vom Clustering-Algorithmus verwendeten Spalten werden von den Parametern coordinate_1, coordinate_2 usw. bis zu n Spalten festgelegt. Es handelt sich bei allen um Aggregierungen. Die Anzahl der erstellten Cluster wird durch den Parameter num_clusters bestimmt.

```
KMeansND - Diagrammfunktion(num_clusters, num_iter, coordinate_1, coordinate_
2 [,coordinate_3 [, ...]])
```

KMeansCentroid2D

KMeansCentroid2D() wertet die Zeilen des Diagramms aus, indem K-means-Clustering angewandt wird. Für jede Diagrammzeile wird die gewünschte Koordinate des Clusters angezeigt, dem dieser Datenpunkt zugewiesen wurde. Die vom Clustering-Algorithmus verwendeten Spalten werden von den Parametern coordinate_1 bzw. coordinate_2 festgelegt. Es handelt sich bei beiden um Aggregierungen. Die Anzahl der erstellten Cluster wird durch den Parameter num_clusters bestimmt. Daten können optional mit dem Normparameter normalisiert werden.

```
KMeansCentroid2D - Diagrammfunktion(num_clusters, coordinate_no, coordinate_
1, coordinate_2 [, norm])
```

KMeansCentroidND

KMeansCentroidND() wertet die Zeilen des Diagramms aus, indem K-means-Clustering angewandt wird. Für jede Diagrammzeile wird die gewünschte Koordinate des Clusters angezeigt, dem dieser Datenpunkt zugewiesen wurde. Die vom Clustering-Algorithmus verwendeten Spalten werden von den Parametern coordinate_1, coordinate_2 usw. bis zu n Spalten festgelegt. Es handelt sich bei allen um Aggregierungen. Die Anzahl der erstellten Cluster wird durch den Parameter num_clusters bestimmt.

```
KMeansCentroidND - Diagrammfunktion(num_clusters, num_iter, coordinate_no,
coordinate_1, coordinate_2 [,coordinate_3 [, ...]])
```

Zeitreihenzerlegungs-Funktionen

STL_Trend

STL_Trend ist eine Zeitreihenzerlegungsfunktion. Zusammen mit **STL_Seasonal** und **STL_Residual** wird diese Funktion verwendet, um eine Zeitreihe in Saison-, Trend- und Restkomponenten zu zerlegen. Im Kontext des STL-Algorithmus wird Zeitreihenzerlegung verwendet, um sowohl Saisonmuster als auch einen allgemeinen Trend aus einer Eingabemetrik und anderen Parametern zu ermitteln. Die Funktion **STL_Trend** identifiziert einen allgemeinen, von Saisonmustern oder -zyklen unabhängigen Trend in den Zeitreihendaten.

```
STL_Trend - Diagrammfunktion(target_measure, period_int [,seasonal_smoother
[,trend_smoother]])
```

STL_Seasonal

STL_Seasonal ist eine Zeitreihenzerlegungsfunktion. Zusammen mit **STL_Trend** und **STL_Residual** wird diese Funktion verwendet, um eine Zeitreihe in Saison-, Trend- und Restkomponenten zu zerlegen. Im Kontext des STL-Algorithmus wird Zeitreihenzerlegung verwendet, um sowohl Saisonmuster als auch einen allgemeinen Trend aus einer Eingabemetrik und anderen Parametern zu ermitteln. Die Funktion **STL_Seasonal** kann ein Saisonmuster in einer Zeitreihe identifizieren und es vom allgemeinen, in den Daten angezeigten Trend trennen.

```
STL_Seasonal - Diagrammfunktion(target_measure, period_int [,seasonal_
smoother [,trend_smoother]])
```

STL_Residual

STL_Residual ist eine Zeitreihenzerlegungsfunktion. Zusammen mit **STL_Seasonal** und **STL_Trend** wird diese Funktion verwendet, um eine Zeitreihe in Saison-, Trend- und Restkomponenten zu zerlegen. Im Kontext des STL-Algorithmus wird Zeitreihenzerlegung verwendet, um sowohl Saisonmuster als auch einen allgemeinen Trend aus einer Eingabemetrik und anderen Parametern zu ermitteln. Wenn diese Operation ausgeführt wird, passen manche der Variationen in der Eingabemetrik weder zur Saison- noch zur Trendkomponente und werden als Restkomponente definiert. Die Diagrammfunktion **STL_Residual** erfasst diesen Teil der Berechnung.

```
STL_Residual - Diagrammfunktion(target_measure, period_int [,seasonal_
smoother [,trend_smoother]])
```

Rank - Diagrammfunktion

Rank() berechnet die Zeilen des Diagramms in der Formel und zeigt für jede Zeile die relative Position des Wertes der Dimension an, die in der Formel berechnet wird. Beim Auswerten der Formel vergleicht die Funktion ihr Ergebnis mit den Ergebnissen für andere Zeilen innerhalb des Spaltenabschnitts und liefert den Rang der Zeile innerhalb des Spaltenabschnitts.

Spaltensegmente

Region	Country	Population	Rank(Population)
Americas	Mexico	128,932,753	2
Americas	Canada	37,742,154	3
Americas	United States of America	331,002,651	1
Europe	Sweden	10,099,265	4
Europe	United Kingdom	67,886,011	2
Europe	France	65,273,511	3
Europe	Germany	83,783,942	1

Bei anderen Diagrammen als Tabellen wird der aktuelle Spaltenabschnitt so definiert, wie er im entsprechenden Tabellendiagramm des Diagramms erscheint.

Syntax:

```
Rank([TOTAL] expr[, mode[, fmt]])
```

Rückgabe Datentyp: dual

Argumente:

Argumente

Argument	Beschreibung
expr	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.
mode	Definiert die numerische Repräsentation des Rangs.
fmt	Definiert die Textrepräsentation des Rangs.
TOTAL	Wenn das Diagramm nur eine Dimension hat oder die Formel mit dem Qualifizierer **TOTAL** versehen ist, wird die Funktion über die gesamten Spalte ausgewertet. Hat das Diagramm mehrere vertikale Dimensionen, so umfasst der aktuelle Spaltenabschnitt nur Zeilen mit denselben Werten wie in der aktuellen Zeile in allen Dimensionsspalten, mit Ausnahme der in der Priorität der Sortierfolgen letzten Dimension.

Der Rang ist eine duale Größe. Hat jede Spalte einen eindeutigen Rang, wird dieser als ganze Zahl zwischen 1 und der Zahl der Zeilen des aktuellen Zeilenabschnitts ausgegeben.

Teilen sich mehrere Zeilen denselben Rang, kann das Ergebnis der Funktion durch die Parameter **mode** und **fmt** modifiziert werden.

mode

Das zweite Argument **mode** kann folgende Werte annehmen:

mode-Beispiele

Wert	Beschreibung
0 (Standard)	Fallen die Ränge alle unterhalb des mittleren Ranges der gesamten Rangfolge, erhalten alle Zeilen der Gruppe den geringsten innerhalb dieser Gruppe auftretenden Rang. Fallen die Ränge alle oberhalb des mittleren Ranges der gesamten Rangfolge, erhalten alle Zeilen der Gruppe den höchsten innerhalb dieser Gruppe auftretenden Rang. Geht die Gruppe der Zeilen gleichen Rangs über den mittleren Rang der gesamten Rangfolge hinweg, erhalten alle Zeilen der Gruppe den mittleren Rang der gesamten Rangfolge.
1	Alle Zeilen erhalten den niedrigsten Rang.
2	Alle Zeilen erhalten den mittleren Rang.
3	Alle Zeilen erhalten den höchsten Rang.
4	Die erste Zeile erhält den niedrigsten Rang, für die Ränge nachfolgender Zeilen wird jeweils eins addiert.

fmt

Das dritte Argument **fmt** kann folgende Werte annehmen:

fmt-Beispiele

Wert	Beschreibung
0 (Standard)	Niedrigster Wert - höchster Wert in allen Zeilen (z. B. 3 - 4).
1	Niedrigster Wert in allen Zeilen.
2	Niedrigster Wert in der ersten Zeile, leere Zeilen in allen weiteren Zeilen.

Die für **mode** 4 und **fmt** 2 maßgebliche Reihenfolge ist die Sortierfolge der Diagrammdimension.

Beispiele und Ergebnisse:

Erstellen Sie zwei Visualisierungen aus den Dimensionen Product und Sales und eine weitere aus Product und UnitSales. Fügen Sie, wie in der folgenden Tabelle gezeigt, Kennzahlen hinzu.

Beispiele für Rangfolge

Beispiele	Ergebnisse
Beispiel 1. Erstellen Sie eine Tabelle mit den Dimensionen `Customer` und `Sales` und der Kennzahl `Rank(Sales)`.	Das Ergebnis hängt von der Sortierreihenfolge der Dimensionen ab. Wenn die Tabelle nach Customer sortiert wird, werden in der Tabelle alle Werte für Sales fürAstrida, danach fürBetacab usw. aufgeführt. Die Ergebnisse für Rank(Sales) zeigen 10 für den Wert Sales 12, 9 für den Wert Sales 13 usw. an, wobei sich der Rangwert 1 für den Wert Sales 78 ergibt. Der nächste Spaltenabschnitt beginnt mit Betacab, für den der erste Wert von Sales im Segment 12 ist. Für den Rangwert von Rank(Sales) ergibt sich 11. Wird die Tabelle nach Sales sortiert, bestehen die Spaltenabschnitte aus den Werten von Sales und dem entsprechenden Customer. Weil es zwei Werte Sales 12 gibt (für Astrida und Betacab), ergibt sich für den Wert von Rank(Sales) für jeden Wert von Customer dieses Spaltenabschnitts 1-2. Grund hierfür ist, dass es zwei Werte Customer gibt, bei denen der Wert Sales 12 beträgt. Im Fall von 4 Werten wäre das Ergebnis für alle Zeilen 1-4. Hier wird gezeigt, wie das Ergebnis für den Standardwert (0) des Arguments fmt aussieht.
Beispiel 2. Ersetzen Sie die Dimension „Kunde“ durch „Produkt“ und fügen Sie die Kennzahl `Rank(Sales,1,2)` hinzu.	Dadurch ergibt sich für die erste Zeile jedes Spaltenabschnitts 1 und alle anderen Zeilen bleiben leer, da für die Argumente **mode** und **fmt** 1 bzw. 2 gewählt wurde.

Ergebnis aus Beispiel 1, Tabelle sortiert nach Customer:

Ergebnistabelle

Customer	Sales	Rank(Sales)
Astrida	12	10
Astrida	13	9
Astrida	20	8
Astrida	22	7
Astrida	45	6
Astrida	46	5
Astrida	60	4
Astrida	65	3
Astrida	70	2
Astrida	78	1
Betcab	12	11

Ergebnis aus Beispiel 1, Tabelle sortiert nach Sales:

Ergebnistabelle

Customer	Sales	Rank(Sales)
Astrida	12	1-2
Betacab	12	1-2
Astrida	13	1
Betacab	15	1
Astrida	20	1
Astrida	22	1-2
Betacab	22	1-2
Betacab	24	1-2
Canutility	24	1-2

In Beispielen verwendete Daten:

```
ProductData:

Load * inline [

Customer|Product|UnitSales|UnitPrice

Astrida|AA|4|16
```

```
Astrida|AA|10|15

Astrida|BB|9|9

Betacab|BB|5|10

Betacab|CC|2|20

Betacab|DD|0|25

Canutility|AA|8|15

Canutility|CC|0|19

] (delimiter is '|');


Sales2013:
crosstable (Month, Sales) LOAD * inline [
Customer|Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec
Astrida|46|60|70|13|78|20|45|65|78|12|78|22
Betacab|65|56|22|79|12|56|45|24|32|78|55|15
Canutility|77|68|34|91|24|68|57|36|44|90|67|27
Divadip|57|36|44|90|67|27|57|68|47|90|80|94
] (delimiter is '|');
```

Siehe auch:

Sum - Diagrammfunktion (page 363)

HRank - Diagrammfunktion

HRank() wertet die Formel aus und vergleicht ihr Ergebnis mit den Ergebnissen für andere Spalten desselben Zeilenabschnitts der Pivottabelle. Die Funktion liefert anschließend die Rangfolge der Spalte innerhalb des Abschnitts.

Syntax:

```
HRank([ TOTAL ] expr [ , mode [, fmt ] ])
```

Rückgabe Datentyp: dual

Diese Funktion ist ausschließlich für Pivottabellen vorgesehen. In allen anderen Diagrammtypen liefert sie NULL.

Argumente:

Argumente

Argument	Beschreibung
expr	Die Formel oder das Feld mit den Daten, die gemessen werden sollen.
mode	Definiert die numerische Repräsentation des Rangs.
fmt	Definiert die Textrepräsentation des Rangs.
TOTAL	Wenn das Diagramm nur eine Dimension hat oder die Formel mit dem Qualifizierer **TOTAL** versehen ist, wird die Funktion über die gesamten Spalte ausgewertet. Hat das Diagramm mehrere vertikale Dimensionen, so umfasst der aktuelle Spaltenabschnitt nur Zeilen mit denselben Werten wie in der aktuellen Zeile in allen Dimensionsspalten, mit Ausnahme der in der Priorität der Sortierfolgen letzten Dimension.

Wenn die Pivottabelle nur eine Dimension hat oder die Formel mit dem Zusatz **total** versehen ist, entspricht der Zeilenabschnitt der gesamten Zeile. Hat die Pivottabelle dagegen mehrere horizontale Dimensionen, so umfasst der Zeilenabschnitt nur Spalten, deren Werte in allen Dimensionen mit Ausnahme der in der Priorität der Sortierfolgen letzten Dimension übereinstimmen.

Der Rang ist eine duale Größe. Hat jede Spalte einen eindeutigen Rang, wird dieser als ganze Zahl zwischen 1 und der Zahl der Spalten des Zeilenabschnitts ausgegeben.

Teilen sich mehrere Spalten denselben Rang, kann das Ergebnis der Funktion durch die Argumente **mode** und **format** modifiziert werden.

Der zweite Parameter **mode** definiert die numerische Repräsentation des Rangs:

mode-Beispiele

Wert	Beschreibung
0 (Standard)	Fallen die Ränge alle unterhalb des mittleren Ranges der gesamten Rangfolge, erhalten alle Spalten der Gruppe den geringsten innerhalb dieser Gruppe auftretenden Rang. Fallen die Ränge alle oberhalb des mittleren Ranges der gesamten Rangfolge, erhalten alle Spalten der Gruppe den höchsten innerhalb dieser Gruppe auftretenden Rang. Geht die Gruppe der Zeilen gleichen Rangs über den mittleren Rang der gesamten Rangfolge hinweg, erhalten alle Zeilen der Gruppe den mittleren Rang der gesamten Rangfolge.
1	Alle Spalten erhalten den niedrigsten Rang innerhalb der Gruppe.
2	Alle Spalten erhalten den mittleren Rang innerhalb der Gruppe.
3	Alle Spalten erhalten den höchsten Rang innerhalb der Gruppe.
4	Die erste Spalte der Gruppe erhält den niedrigsten Rang, für die Ränge nachfolgender Spalten wird jeweils 1 addiert.

Das dritte Argument **format** definiert die Textrepräsentation des Rangs:

format-Beispiele

Wert	Beschreibung
0 (Standard)	Niedrigster Wert**&' - '&**höchster Wert in allen Spalten der Gruppe (z. B. 3 - 4)
1	Niedrigster Wert in allen Spalten der Gruppe.
2	Niedrigster Wert in der ersten Spalte, leere Zellen in allen weiteren Spalten der Gruppe.

Die für **mode** 4 und **format** 2 maßgebliche Reihenfolge ist die Sortierfolge der Diagrammdimension.

Beispiele:

```
HRank( sum( Sales ))
```

```
HRank( sum( Sales ), 2 )
```

```
HRank( sum( Sales ), 0, 1 )
```

Optimieren mit k-means: Ein reales Beispiel

Das folgende reale Beispiel zeigt einen Fall, in dem KMeans-Clustering und Zentroid-Funktionen auf einen Datensatz angewendet werden. Die KMeans-Funktion teilt Datenpunkte in Cluster auf, die Ähnlichkeiten teilen. Die Cluster werden kompakter und differenzierter, während der KMeans-Algorithmus über eine konfigurierbare Anzahl Iterationen angewendet wird.

KMeans wird in vielen Gebieten für unterschiedliche Anwendungsfälle eingesetzt. Einige Beispiele für Clustering-Anwendungsfälle sind Kundensegmentierung, Betrugserkennung, Prognose der Kundenabwanderung, Targeting von Kundenanreizen, Identifizierung von Cyberkriminalität und Optimierung von Lieferstrecken. Der KMeans-Clustering-Algorithmus wird immer häufiger verwendet, wenn Unternehmen versuchen, Muster abzuleiten und Dienstangebote zu optimieren.

Qlik Sense KMeans- und Zentroid-Funktionen

Qlik Sense stellt zwei KMeans-Funktionen bereit, die Datenpunkte in Clustern basierend auf ihrer Ähnlichkeit gruppieren. Weitere Informationen finden Sie unter *KMeans2D - Diagrammfunktion (page 1419)* und *KMeansND - Diagrammfunktion (page 1434)*. Die **KMeans2D**-Funktion akzeptiert zwei Dimensionen und funktioniert gut beim Visualisieren von Ergebnissen über ein **Punktdiagramm**. Die **KMeansND**-Funktion akzeptiert mehr als zwei Dimensionen. Da sich die Konzepte eines 2D-Ergebnisses leicht in Standarddiagrammen darstellen lassen, wird in der folgenden Demo KMeans auf ein **Punktdiagramm** unter Verwendung von zwei Dimensionen angewandt. KMeans-Clustering kann durch Einfärbung nach Formel oder wie in diesem Beispiel beschrieben durch Dimensionen visualisiert werden.

Qlik Sense Zentroid-Funktionen bestimmen die Position des arithmetischen Mittels aller Datenpunkte im Cluster und identifizieren einen zentralen Punkt bzw. Zentroid für diesen Cluster. Für jede Diagrammzeile (bzw. jeden Datensatz) zeigt die Zentroid-Funktion die Koordinate des Clusters an, der dieser Datenpunkt zugewiesen wurde. Weitere Informationen finden Sie unter *KMeansCentroid2D - Diagrammfunktion (page 1449)* und *KMeansCentroidND - Diagrammfunktion (page 1450)*.

Anwendungsfall und Beispiel – Übersicht

Das folgende Beispiel führt durch die Schritte eines simulierten realen Szenarios. Ein Textilunternehmen im Bundesstaat New York, USA, muss seine Ausgaben senken, indem es die Lieferkosten minimiert. Eine Möglichkeit besteht darin, Lagerstandorte zu verlegen, damit sie sich näher bei den Distributoren befinden. Das Unternehmen beschäftigt 118 Distributoren im Bundesstaat New York. Die folgende Demo simuliert, wie ein Operations Manager die Distributoren mit der KMeans-Funktion in fünf geclusterte geografische Gebiete segmentieren könnte. Anschließend identifiziert er anhand der Zentroid-Funktion fünf optimale Lagerstandorte, die für diese Cluster zentral liegen. Das Ziel besteht darin, Zuordnungskoordinaten zu finden, die zum Identifizieren von fünf zentralen Lagerstandorten verwendet werden können.

Der Datensatz

Der Datensatz basiert auf zufällig generierten Namen und Adressen im Bundesstaat New York mit echten Längen- und Breitenkoordinaten. Der Datensatz enthält die folgenden zehn Spalten: id, first_name, last_name, telephone, address, city, state, zip, latitude, longitude. Der Datensatz steht unten als Datei zur Verfügung, die Sie lokal herunterladen und dann an Qlik Sense hochladen oder inline im Dateneditor verwenden können. Die erstellte App erhält den Namen *Distributors KMeans and Centroid*, und das erste Arbeitsblatt in der App wird *Distribution cluster analysis* genannt.

Klicken Sie auf den folgenden Link, um die Beispieldatendatei herunterzuladen: DistributorData.csv

Distributor-Datensatz: Inline-Ladevorgang für Dateneditor in Qlik Sense (page 1417)

Title: DistributorData

Gesamtzahl Datensätze: 118

Anwenden der KMeans2D-Funktion

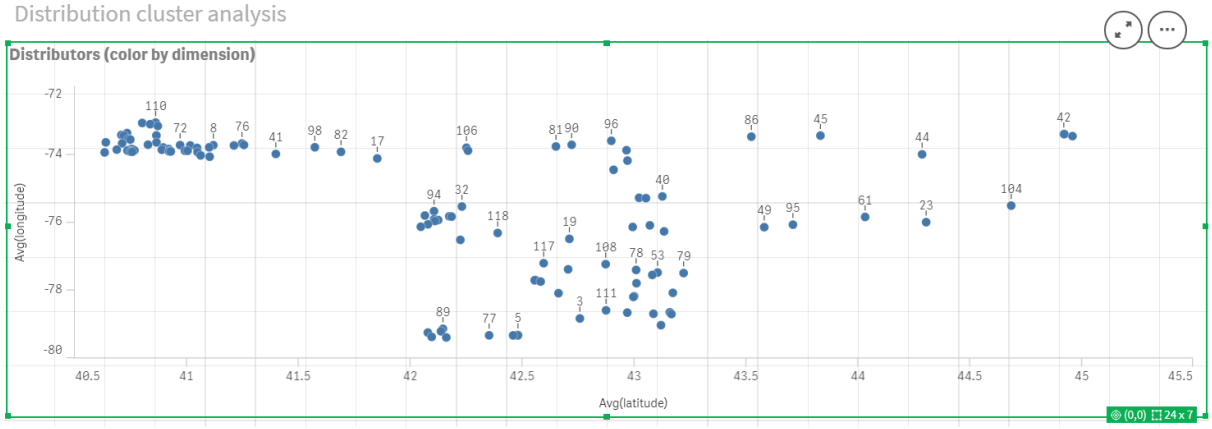
In diesem Beispiel wird die Konfiguration eines **Punktdiagramms** unter Verwendung des Datensatzes *DistributorData* gezeigt, die **KMeans2D**-Funktion wird angewendet, und das Diagramm wird nach Dimension eingefärbt.

Beachten Sie, dass Qlik Sense KMeans-Funktionen automatisches Clustering mit einer Methode unterstützen, die als Tiefendifferenz bezeichnet wird. Wenn ein Benutzer 0 für die Anzahl der Cluster festlegt, wird die optimale Anzahl Cluster für diesen Datensatz bestimmt. Für dieses Beispiel wird aber eine Variable für das Argument **num_clusters** erstellt (die Syntax entnehmen Sie *KMeans2D - Diagrammfunktion (page 1419)*). Daher wird die gewünschte Anzahl Cluster (k=5) von einer Variablen angegeben.

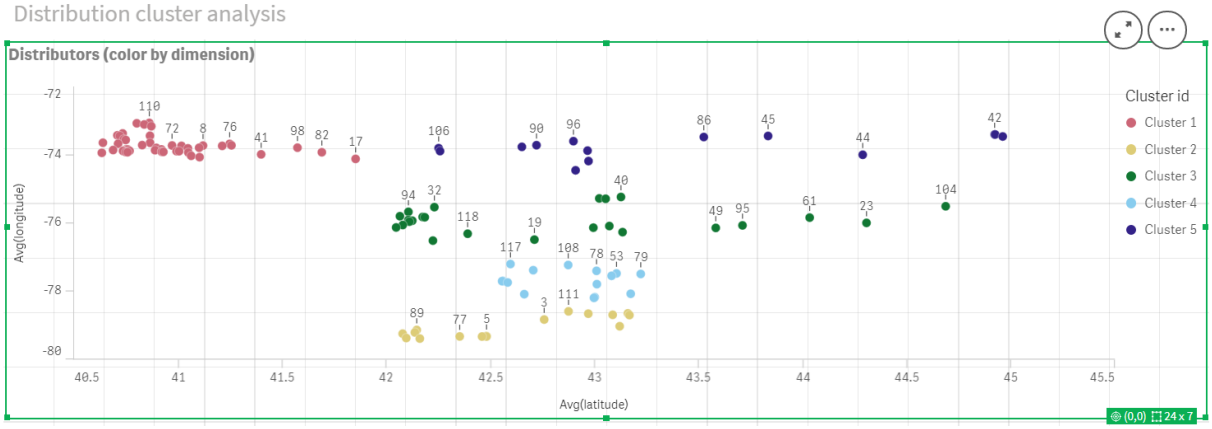
1. Ein **Punktdiagramm** wird auf das Arbeitsblatt gezogen und erhält den Namen *Distributors (by dimension)*.
2. Eine **Variable** wird erstellt, um die Anzahl der Cluster anzugeben. Die **Variable** erhält den Namen *vDistClusters*. Geben Sie für die Variable **Definition** den Wert *5* ein.
3. **Daten**-Konfiguration für das Diagramm:
    a. Unter **Dimensionen** wird das Feld *ID* für **Blase** ausgewählt. *Cluster-ID* wird für die **Bezeichnung** eingegeben.
    b. Unter **Kennzahlen** ist *Avg([latitude])* die Formel für die **X-Achse**.
    c. Unter **Kennzahlen** ist *Avg([longitued])* die Formel für die **Y-Achse**.

4. Konfiguration der **Darstellung**:
    a. Unter **Farben und Legenden** wird die Option **Benutzerdefiniert** für **Farben** ausgewählt.
    b. **Nach Dimension** wird für das Einfärben des Diagramms ausgewählt.
    c. Die folgende Formel wird eingegeben: *=pick(aggr(KMeans2D(vDistClusters,only(latitude),only (longitude)),id)+1, 'Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster 4', 'Cluster 5')*
    d. Das Kontrollkästchen für **Farben bei Auswahl beibehalten** wird aktiviert.

Punktdiagramm vor Anwenden der KMeans-Einfärbung nach Dimension



Punktdiagramm nach Anwenden der KMean-Einfärbung nach Dimension



Hinzufügen einer **Tabelle**: *Distributors*

Es kann sinnvoll sein, eine Tabelle für den schnellen Zugriff auf relevante Daten zur Hand zu haben. Das **Punktdiagramm** zeigt *IDs* über eine Tabelle; die zugehörigen Distributornamen wurden als Referenz hinzugefügt.

1. Eine **Tabelle** mit dem Namen *Distributors* wird auf das Arbeitsblatt gezogen, der die folgenden **Spalten** (Dimensionen) hinzugefügt werden: *id*, *first_name* und *last_name*.

Tabelle: Distributor names

Distributors

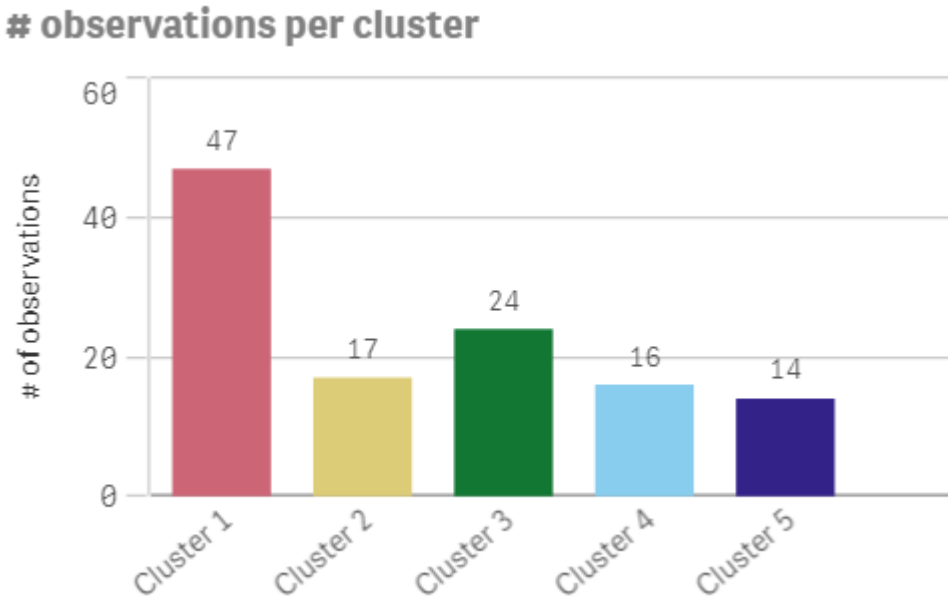
id	first_name	last_name
1	Kaiya	Snow
2	Dean	Roy
3	Eden	Paul
4	Bryanna	Higgins
5	Elisabeth	Lee
6	Skylar	Robinson
7	Cody	Bailey
8	Dario	Sims
9	Deacon	Hood

Hinzufügen eines **Balkendiagramms**: *# observations per cluster*

Für das Szenario der Lagerverteilung ist es sinnvoll zu wissen, wie viele Distributoren von jedem Lager beliefert werden. Daher wird ein **Balkendiagramm** erstellt, das misst, wie viele Distributoren jedem Cluster hinzugefügt werden.

1. Ein **Balkendiagramm** wird auf das Arbeitsblatt gezogen. Das Diagramm erhält den Namen *# observations per cluster*.
2. **Daten**-Konfiguration für das **Balkendiagramm**:
    a. Eine **Dimension** mit der Bezeichnung *Clusters* wird hinzugefügt (die Bezeichnung kann hinzugefügt werden, nachdem die Formel angewendet wurde). Die folgende Formel wird eingegeben: *=pick(aggr(KMeans2D(vDistClusters,only(latitude),only(longitude)),id)+1, 'Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster 4', 'Cluster 5')*
    b. Eine **Kennzahl** mit der Bezeichnung *# observations* wird hinzugefügt. Die folgende Formel wird eingegeben: *=count(aggr(KMeans2D(vDistClusters,only(latitude),only(longitude)),id))*
3. Konfiguration der **Darstellung**:
    a. Unter **Farben und Legenden** wird die Option **Benutzerdefiniert** für **Farben** ausgewählt.
    b. **Nach Dimension** wird für das Einfärben des Diagramms ausgewählt.
    c. Die folgende Formel wird eingegeben: *=pick(aggr(KMeans2D(vDistClusters,only(latitude),only(longitude)),id)+1, 'Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster 4', 'Cluster 5')*
    d. Das Kontrollkästchen für **Farben bei Auswahl beibehalten** wird aktiviert.
    e. **Legende anzeigen** ist deaktiviert.
    f. Unter **Präsentation** ist **Wertbezeichnungen** auf **Automatisch** festgelegt.
    g. Unter **x-Achse**: **Clusters** ist **Nur Feldnamen** ausgewählt.

Balkendiagramm: # observations per cluster



Anwenden der **Centroid2D**-Funktion

Eine zweite Tabelle wird für die **Centroid2D**-Funktion hinzugefügt, mit der die Koordinaten für mögliche Lagerstandorte identifiziert werden. Diese Tabelle zeigt den zentralen Standort (Zentroid-Werte) für die fünf identifizierten Distributorgruppen.

1. Eine **Tabelle** wird auf das Arbeitsblatt gezogen und erhält den Namen *Cluster centroids*. Ihr wurden folgende Spalten hinzugefügt:
    a. Eine **Dimension** mit der Bezeichnung *Clusters* wird hinzugefügt. Die folgende Formel wird eingegeben:=*pick(aggr(KMeans2D(vDistClusters,only(latitude),only(longitude)),id)+1,'Warehouse 1','Warehouse 2','Warehouse 3','Warehouse 4','Warehouse 5'*)
    b. Eine **Kennzahl** mit der Bezeichnung *latitude (D1)* wird hinzugefügt. Die folgende Formel wird eingegeben:=*only(aggr(KMeansCentroid2D(vDistClusters,0,only(latitude),only(longitude)),id))* Beachten Sie, dass der Parameter **coordinate_no** der ersten Dimension (0) entspricht. In diesem Fall wird die Dimension *latitude* gegen die x-Achse aufgetragen. Wenn mit der **CentroidND**-Funktion gearbeitet würde und bis zu sechs Dimensionen vorhanden wären, könnten diese Parametereinträge einen beliebigen der sechs Werte 0, 1, 2, 3, 4 oder 5 haben.
    c. Eine **Kennzahl** mit der Bezeichnung *longitude (D2)* wird hinzugefügt. Die folgende Formel wird eingegeben: =*only(aggr(KMeansCentroid2D(vDistClusters,1,only(latitude),only(longitude)),id))*D Der Parameter **coordinate_no** in dieser Formel entspricht der zweiten Dimension (1). Die Dimension *longitude* wird gegen die y-Achse aufgetragen.

Tabelle: Cluster-Zentroid-Berechnungen

Cluster centroids

Clusters	latitude (D1)	longitude (D2)
Totals	-	-
Warehouse 1	40.945422240426	-73.719966482979
Warehouse 2	42.590538729412	-79.067889217647
Warehouse 3	42.805089516667	-75.901621883333
Warehouse 4	42.8581692625	-77.6800485875
Warehouse 5	43.436770771429	-73.734622635714

Zentroid-Zuordnung

Im nächsten Schritt werden die Zentroide zugeordnet. Der App-Entwickler kann entscheiden, ob die Visualisierung auf getrennten Arbeitsblättern platziert werden soll.

1. Eine **Karte** mit dem Namen *Centroid mapping* wird auf das Arbeitsblatt gezogen.
2. Im Abschnitt **Ebenen** wird **Ebene hinzufügen** und dann **Punktebene** ausgewählt.
    a. Das **Feld** *id* wird ausgewählt, und die *Bezeichnung* **Dist ids** wird hinzugefügt.
    b. Im Abschnitt **Standort** wird das Kontrollkästchen für **Felder für Längen- und Breitengrad** aktiviert.
    c. Für **Breitengrad** wird das Feld *latitude* ausgewählt.
    d. Für **Längengrad** wird das Feld *longitude* ausgewählt.
    e. Im Abschnitt **Größe und Form** wird **Blase** für **Form** ausgewählt und die **Größe** wird nach Wunsch mit dem Schieberegler verringert.
    f. Im Abschnitt **Farben** wird **Eine Farbe** ausgewählt, und Blau wird für die **Farbe** und Grau für den **Umriss** ausgewählt. (Diese Auswahlen können nach Wunsch geändert werden.)
3. Im Abschnitt **Ebenen** wird eine zweite **Punktebene** hinzugefügt, indem **Ebene hinzufügen** und dann **Punktebene** ausgewählt wird.
    a. Die folgende Formel wird eingegeben: *=aggr(KMeans2D(vDistClusters,only(latitude),only (longitude)),id)*
    b. Die **Bezeichnung** *Cluster* wird hinzugefügt.
    c. Im Abschnitt **Standort** wird das Kontrollkästchen für **Felder für Längen- und Breitengrad** aktiviert.
    d. Für die **Breite**, die in diesem Fall entlang der x-Achse aufgetragen wird, wird folgende Formel hinzugefügt: *=aggr(KMeansCentroid2D(vDistClusters,0,only(latitude),only(longitude)),id)*
    e. Für die **Länge**, die in diesem Fall entlang der y-Achse aufgetragen wird, wird folgende Formel hinzugefügt: *=aggr(KMeansCentroid2D(vDistClusters,1,only(latitude),only(longitude)),id)*
    f. Im Abschnitt **Größe und Form** wird **Dreieck** für **Form** ausgewählt und die **Größe** wird nach Wunsch mit dem Schieberegler verringert.

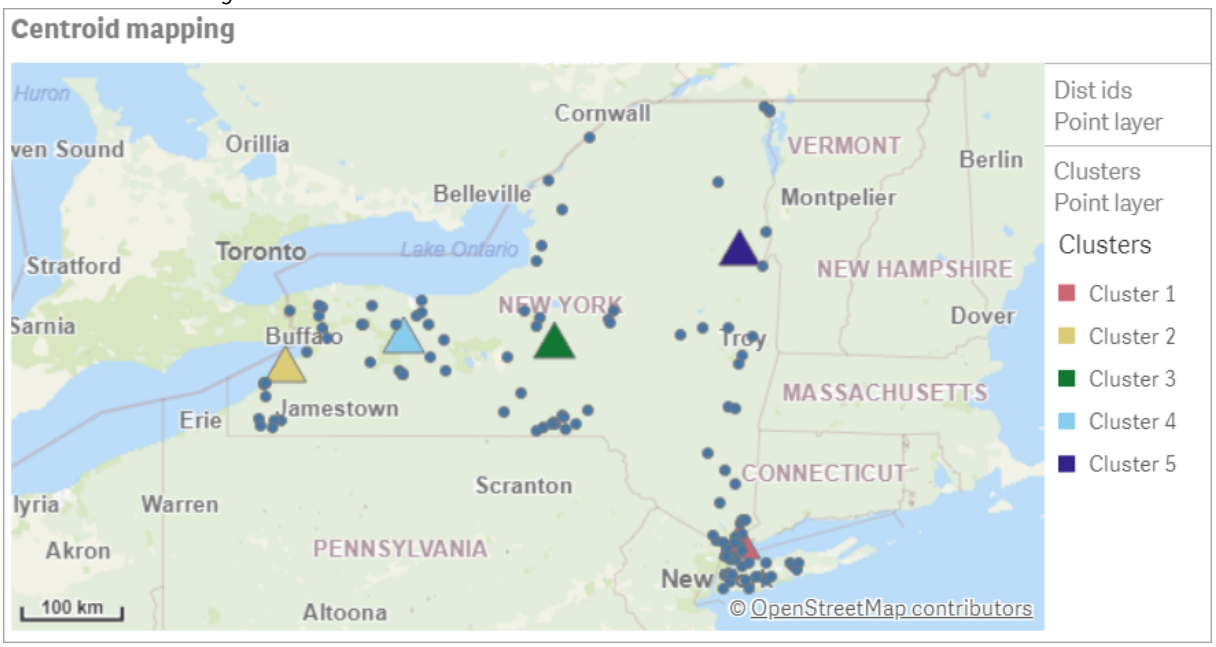
g. Unter **Farben und Legenden** wird die Option **Benutzerdefiniert** für **Farben** ausgewählt.

h. **Nach Dimension** wird für das Einfärben des Diagramms ausgewählt. Die folgende Formel wird eingegeben: *=pick(aggr(KMeans2D(vDistClusters,only(latitude),only(longitude)),id)+1,'Cluster 1','Cluster 2','Cluster 3','Cluster 4','Cluster 5')*

i. Die Dimension erhält die Bezeichnung *Clusters*.

4. In **Karteneinstellungen** wird **Adaptiv** für **Projektion** ausgewählt. **Metrisch** wird für **Maßeinheiten** ausgewählt.

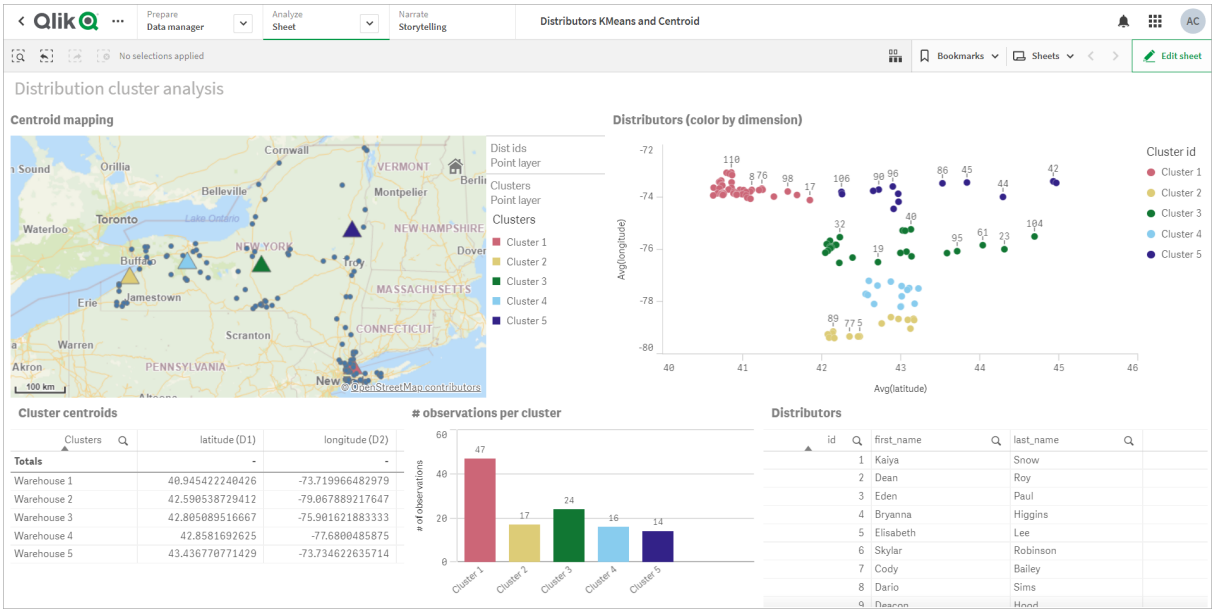
Karte: Nach Cluster zugeordnete Zentroide



Zusammenfassung

Anhand der KMeans-Funktion wurden in diesem realen Szenario Distributoren in ähnliche Gruppen oder Cluster gestützt auf Ähnlichkeit segmentiert; in diesem Fall war dies die Nähe zueinander. Die Zentroid-Funktion wurde auf diese Cluster angewendet, um fünf Zuordnungskoordinaten zu identifizieren. Diese Koordinaten stellen einen anfänglichen zentralen Standort bereit, an dem Lager gebaut oder untergebracht werden können. Die Zentroid-Funktion wird auf das **Karten**-Diagramm angewendet, sodass App-Benutzer visualisieren können, wo sich die Zentroide relativ zu umgebenden Cluster-Datenpunkten befinden. Die sich daraus ergebenden Koordinaten stellen mögliche Lagerstandorte dar, mit denen die Lieferkosten an Distributoren im Bundesstaat New York minimiert werden könnten.

App: Beispiel für KMeans- und Zentroid-Analyse



Distributor-Datensatz: Inline-Ladevorgang für Dateneditor in Qlik Sense

```
DistributorData:
Load * Inline [
id,first_name,last_name,telephone,address,city,state,zip,latitude,longitude
1,Kaiya,Snow,(716) 201-1212,6231 Tonawanda Creek Rd #APT 308,Lockport,NY,14094,43.08926,-
78.69313
2,Dean,Roy,(716) 201-1588,6884 E High St,Lockport,NY,14094,43.16245,-78.65036
3,Eden,Paul,(716) 202-4596,4647 Southwestern Blvd #APT 350,Hamburg,NY,14075,42.76003,-78.83194
4,Bryanna,Higgins,(716) 203-7041,418 Park Ave,Dunkirk,NY,14048,42.48279,-79.33088
5,Elisabeth,Lee,(716) 203-7043,36 E Courtney St,Dunkirk,NY,14048,42.48299,-79.31928
6,Skylar,Robinson,(716) 203-7166,26 Greco Ln,Dunkirk,NY,14048,42.4612095,-79.3317925
7,Cody,Bailey,(716) 203-7201,114 Lincoln Ave,Dunkirk,NY,14048,42.4801269,-79.322232
8,Dario,Sims,(408) 927-1606,N Castle Dr,Armonk,NY,10504,41.11979,-73.714864
9,Deacon,Hood,(410) 244-6221,4856 44th St,Woodside,NY,11377,40.748372,-73.905445
10,Zackery,Levy,(410) 363-8874,61 Executive Blvd,Farmingdale,NY,11735,40.7197457,-73.430239
11,Rey,Hawkins,(412) 344-8687,4585 Shimerville Rd,Clarence,NY,14031,42.972075,-78.6592452
12,Phillip,Howard,(413) 269-4049,464 Main St #101,Port Washington,NY,11050,40.8273756,-
73.7009971
13,Shirley,Tyler,(434) 985-8943,114 Glann Rd,Apalachin,NY,13732,42.0482515,-76.1229725
14,Aniyah,Jarvis,(440) 244-1808,87 N Middletown Rd,Pearl River,NY,10965,41.0629,-74.0159
15,Alayna,Woodard,(478) 335-3704,70 W Red Oak Ln,West Harrison,NY,10604,41.0162722,-73.7234926
16,Jermaine,Lambert,(508) 561-9836,24 Kellogg Rd,New Hartford,NY,13413,43.0555739,-75.2793197
17,Harper,Gibbs,(239) 466-0238,Po Box 33,Cottekill,NY,12419,41.853392,-74.106082
18,Osvaldo,Graham,(252) 246-0816,6878 Sand Hill Rd,East Syracuse,NY,13057,43.073215,-76.081448
19,Roberto,Wade,(270) 469-1211,3936 Holley Rd,Moravia,NY,13118,42.713044,-76.481227
20,Kate,Mcguire,(270) 788-3080,6451 State 64 Rte #3,Naples,NY,14512,42.707366,-77.380489
21,Dale,Andersen,(281) 480-5690,205 W Service Rd,Champlain,NY,12919,44.9645392,-73.4470831
22,Lorelai,Burch,(302) 644-2133,1 Brewster St,Glen Cove,NY,11542,40.865177,-73.633019
23,Amiyah,Flowers,(303) 223-0055,46600 Us Interstate 81 Rte,Alexandria
Bay,NY,13607,44.309626,-75.988365
```

```
24,Mckinley,Clements,(303) 918-3230,200 Summit Lake Dr,Valhalla,NY,10595,41.101145,-73.778298
25,Marc,Gibson,(607) 203-1233,25 Robinson St,Binghamton,NY,13901,42.107416,-75.901614
26,Kali,Norman,(607) 203-1400,1 Ely Park Blvd #APT 15,Binghamton,NY,13905,42.125866,-75.925026
27,Laci,Cain,(607) 203-1437,16 Zimmer Road,Kirkwood,NY,13795,42.066516,-75.792627
28,Mohammad,Perez,(607) 203-1652,71 Endicott Ave #APT 12,Johnson City,NY,13790,42.111894,-
75.952187
29,Izabelle,Pham,(607) 204-0392,434 State 369 Rte,Port Crane,NY,13833,42.185838,-75.823074
30,Kiley,Mays,(607) 204-0870,244 Ballyhack Rd #14,Port Crane,NY,13833,42.175612,-75.814917
31,Peter,Trevino,(607) 205-1374,125 Melbourne St.,Vestal,NY,13850,42.080254,-76.051124
32,Ani,Francis,(607) 208-4067,48 Caswell St,Afton,NY,13730,42.232065,-75.525674
33,Jared,Sheppard,(716) 386-3002,4709 430th Rte,Bemus Point,NY,14712,42.162175,-79.39176
34,Dulce,Atkinson,(914) 576-2266,501 Pelham Rd,New Rochelle,NY,10805,40.895449,-73.782602
35,Jayla,Beasley,(716) 526-1054,5010 474th Rte,Ashville,NY,14710,42.096859,-79.375561
36,Dane,Donovan,(718) 545-3732,5014 31st Ave,Woodside,NY,11377,40.756967,-73.909506
37,Brendon,Clay,(585) 322-7780,133 Cummings Ave,Gainesville,NY,14066,42.664309,-78.085651
38,Asia,Nunez,(718) 426-1472,2407 Gilmore ,East Elmhurst,NY,11369,40.766662,-73.869185
39,Dawson,Odonnell,(718) 342-2179,5019 H Ave,Brooklyn,NY,11234,40.633245,-73.927591
40,Kyle,Collins,(315) 733-7078,502 Rockhaven Rd,Utica,NY,13502,43.129184,-75.226726
41,Eliza,Hardin,(315) 331-8072,502 Sladen Place,West Point,NY,10996,41.3993,-73.973003
42,Kasen,Klein,(518) 298-4581,2407 Lake Shore Rd,Chazy,NY,12921,44.925561,-73.387373
43,Reuben,Bradford,(518) 298-4581,33 Lake Flats Dr,Champlain,NY,12919,44.928092,-73.387884
44,Henry,Grimes,(518) 523-3990,2407 Main St,Lake Placid,NY,12946,44.291487,-73.98474
45,Kyan,Livingston,(518) 585-7364,241 Alexandria Ave,Ticonderoga,NY,12883,43.836553,-73.43155
46,Kaitlyn,Short,(516) 678-3189,241 Chance Dr,Oceanside,NY,11572,40.638534,-73.63079
47,Damaris,Jacobs,(914) 664-5331,241 Claremont Ave,Mount Vernon,NY,10552,40.919852,-73.827848
48,Alivia,Schroeder,(315) 469-4473,241 Lafayette Rd,Syracuse,NY,13205,42.996446,-76.12957
49,Bridget,Strong,(315) 298-4355,241 Maltby Rd,Pulaski,NY,13142,43.584966,-76.136317
50,Francis,Lee,(585) 201-7021,166 Ross St,Batavia,NY,14020,43.0031502,-78.17487
51,Makaila,Phelps,(585) 201-7422,58 S Main St,Batavia,NY,14020,42.99941,-78.1939285
52,Jazlynn,Stephens,(585) 203-1087,1 Sinclair Dr,Pittsford,NY,14534,43.084157,-77.545452
53,Ryann,Randolph,(585) 203-1519,331 Eaglehead Rd,East Rochester,NY,14445,43.10785,-77.475552
54,Rosa,Baker,(585) 204-4011,42 Ossian St,Dansville,NY,14437,42.560761,-77.70088
55,Marcel,Barry,(585) 204-4013,42 Jefferson St,Dansville,NY,14437,42.557735,-77.702983
56,Dennis,Schmitt,(585) 204-4061,750 Dansville Mount Morris Rd,Dansville,NY,14437,42.584458,-
77.741648
57,Cassandra,Kim,(585) 204-4138,3 Perine Ave APT1,Dansville,NY,14437,42.562865,-77.69661
58,Kolton,Jacobson,(585) 206-5047,4925 Upper Holly Rd,Holley,NY,14470,43.175957,-78.074465
59,Nathanael,Donovan,(718) 393-3501,9604 57th Ave,Corona,NY,11373,40.736077,-73.864858
60,Robert,Frazier,(718) 271-3067,300 56th Ave,Corona,NY,11373,40.735304,-73.873997
61,Jessie,Mora,(315) 405-8991,9607 Forsyth Loop,Watertown,NY,13603,44.036466,-75.833437
62,Martha,Rollins,(347) 242-2642,22 Main St,Corona,NY,11373,40.757727,-73.829331
63,Emely,Townsend,(718) 699-0751,60 Sanford Ave,Corona,NY,11373,40.755466,-73.831029
64,Kylie,Cooley,(347) 561-7149,9608 95th Ave,Ozone Park,NY,11416,40.687564,-73.845715
65,Wendy,Cameron,(585) 571-4185,9608 Union St,Scottsville,NY,14546,43.013327,-77.7907839
66,Kayley,Peterson,(718) 654-5027,961 E 230th St,Bronx,NY,10466,40.889275,-73.850555
67,Camden,Ochoa,(718) 760-8699,59 Vark St,Yonkers,NY,10701,40.929322,-73.89957
68,Priscilla,Castillo,(910) 326-7233,9359 Elm St,Chadwicks,NY,13319,43.024902,-75.26886
69,Dana,Schultz,(913) 322-4580,99 Washington Ave,Hastings on Hudson,NY,10706,40.99265,-
73.879748
70,Blaze,Medina,(914) 207-0015,60 Elliott Ave,Yonkers,NY,10705,40.921498,-73.896682
71,Finnegan,Tucker,(914) 207-0015,90 Hillside Drive,Yonkers,NY,10705,40.922514,-73.892911
72,Pranav,Palmer,(914) 214-8376,5 Bruce Ave,Harrison,NY,10528,40.970916,-73.711493
73,Kolten,Wong,(914) 218-8268,70 Barker St,Mount Kisco,NY,10549,41.211993,-73.723202
74,Jasiah,Vazquez,(914) 231-5199,30 Broadway,Dobbs Ferry,NY,10522,41.004629,-73.879825
75,Lamar,Pierce,(914) 232-0380,68 Ridge Rd,Katonah,NY,10536,41.256662,-73.707964
76,Carla,Coffey,(914) 232-0469,197 Beaver Dam Rd,Katonah,NY,10536,41.247934,-73.664363
```

```
77,Brooklynn,Harmon,(716) 595-3227,8084 Glasgow Rd,Cassadega,NY,14718,42.353861,-79.329558
78,Raquel,Hodges,(585) 398-8125,809 County Road ,Victor,NY,14564,43.011745,-77.398806
79,Jerimiah,Gardner,(585) 787-9127,809 Houston Rd,Webster,NY,14580,43.224204,-77.491353
80,Clarence,Hammond,(720) 746-1619,809 Pierpont Ave,Piermont,NY,10968,41.0491181,-73.918622
81,Rhys,Gill,(518) 427-7887,81 Columbia St,Albany,NY,12210,42.652824,-73.752096
82,Edith,Parrish,(845) 452-7621,81 Glenwood Ave,Poughkeepsie,NY,12603,41.691058,-73.910829
83,Kobe,Mcintosh,(845) 371-1101,81 Heitman Dr,Spring Valley,NY,10977,41.103227,-74.054396
84,Ayden,Waters,(516) 796-2722,81 Kingfisher Rd,Levittown,NY,11756,40.738939,-73.52826
85,Francis,Rogers,(631) 427-7728,81 Knollwood Ave,Huntington,NY,11743,40.864905,-73.426107
86,Jaden,Landry,(716) 496-4038,12839 39th Rte,Chaffee,NY,14030,43.527396,-73.462786
87,Giancarlo,Campos,(518) 885-5717,1284 Saratoga Rd,Ballston Spa,NY,12020,42.968594,-73.862847
88,Eduardo,Contreras,(716) 285-8987,1285 Saunders Sett Rd,Niagara Falls,NY,14305,43.122963,-
79.029274
89,Gabriela,Davidson,(716) 267-3195,1286 Mee Rd,Falconer,NY,14733,42.147339,-79.137976
90,Evangeline,Case,(518) 272-9435,1287 2nd Ave,Watervliet,NY,12189,42.723132,-73.703818
91,Tyrone,Ellison,(518) 843-4691,1287 Midline Rd,Amsterdam,NY,12010,42.9730876,-74.1700608
92,Bryce,Bass,(518) 943-9549,1288 Leeds Athens Rd,Athens,NY,12015,42.259381,-73.876897
93,Londyn,Butler,(518) 922-7095,129 Argersinger Rd,Fultonville,NY,12072,42.910969,-74.441917
94,Graham,Becker,(607) 655-1318,129 Baker Rd,Windsor,NY,13865,42.107271,-75.66408
95,Rolando,Fitzgerald,(315) 465-4166,17164 County 90 Rte,Mannsville,NY,13661,43.713443,-
76.06232
96,Grant,Hoover,(518) 692-8363,1718 County 113 Rte,Schaghticote,NY,12154,42.900648,-73.585036
97,Mark,Goodwin,(631) 584-6761,172 Cambon Ave,Saint James,NY,11780,40.871152,-73.146032
98,Deacon,Cantu,(845) 221-7940,172 Carpenter Rd,Hopewell Junction,NY,12533,41.57388,-73.77609
99,Tristian,Walsh,(516) 997-4750,172 E Cabot Ln,Westbury,NY,11590,40.7480397,-73.54819
100,Abram,Alexander,(631) 588-3817,172 Lorenzo Cir,Ronkonkoma,NY,11779,40.837123,-73.09367
101,Lesly,Bush,(516) 489-3791,172 Nassau Blvd,Garden City,NY,11530,40.71147,-73.660753
102,Pamela,Espinoza,(716) 201-1520,172 Niagara St ,Lockport,NY,14094,43.169871,-78.70093
103,Bryanna,Newton,(914) 328-4332,172 Warren Ave,White Plains,NY,10603,41.047207,-73.79572
104,Marcelo,Schmitt,(315) 393-4432,319 Mansion Ave,Ogdensburg,NY,13669,44.690246,-75.49992
105,Layton,Valenzuela,(631) 676-2113,319 Singingwood Dr,Holbrook,NY,11741,40.801391,-73.058993
106,Roderick,Rocha,(518) 671-6037,319 Warren St,Hudson,NY,12534,42.252527,-73.790629
107,Camryn,Terrell,(315) 635-1680,3192 Olive Dr,Baldinsville,NY,13027,43.136843,-76.260303
108,Summer,Callahan,(585) 394-4195,3192 Smith Road,Canandaigua,NY,14424,42.875457,-77.228039
109,Pierre,Novak,(716) 665-2524,3194 Falconer Kimball Stand Rd,Falconer,NY,14733,42.138439,-
79.211091
110,Kennedi,Fry,(315) 543-2301,32 College Rd,Selden,NY,11784,40.861624,-73.04757
111,Wyatt,Pruitt,(716) 681-4042,277 Ransom Rd,Lancaster ,NY,14086,42.87702,-78.591302
112,Lilly,Jensen,(631) 841-0859,2772 Schliegel Blvd,Amityville,NY,11701,40.708021,-73.413015
113,Tristin,Hardin,(631) 920-0927,278 Fulton Street,West Babylon,NY,11704,40.733578,-73.357321
114,Tanya,Stafford,(716) 484-0771,278 Sampson St,Jamestown,NY,14701,42.0797,-79.247805
115,Paris,Cordova,(607) 589-4857,278 Washburn Rd,Spencer,NY,14883,42.225046,-76.510257
116,Alfonso,Morse,(718) 359-5582,200 Colden St,Flushing,NY,11355,40.750403,-73.822752
117,Maurice,Hooper,(315) 595-6694,4435 Italy Hill Rd,Branchport,NY,14418,42.597957,-77.199267
118,Iris,Wolf,(607) 539-7288,444 Harford Rd,Brooktondale,NY,14817,42.392164,-76.30756
];
```

KMeans2D - Diagrammfunktion

KMeans2D() wertet die Zeilen des Diagramms aus, indem K-means-Clustering angewandt wird. Für jede Diagrammzeile wird die Cluster-ID des Clusters angezeigt, dem dieser Datenpunkt zugewiesen wurde. Die vom Clustering-Algorithmus verwendeten Spalten werden von den Parametern coordinate_1 bzw. coordinate_2 festgelegt. Es handelt sich bei beiden um Aggregierungen. Die Anzahl der erstellten Cluster wird durch den Parameter num_clusters bestimmt. Daten können optional mit dem Normparameter normalisiert werden.

KMeans2D gibt einen Wert pro Datenpunkt zurück. Der zurückgegebene Wert ist ein dualer Wert und ein Ganzzahlwert, der dem Cluster entspricht, dem der jeweilige Datenpunkt zugewiesen wurde.

Syntax:

```
KMeans2D(num_clusters, coordinate_1, coordinate_2 [, norm])
```

Rückgabe Datentyp: dual

Argumente:

Argumente

Argument	Beschreibung
num_clusters	Ganze Zahl, die die Anzahl der Cluster angibt
coordinate_1	Die Aggregierung, die die erste Koordinate berechnet, in der Regel die x-Achse des Punktdiagramms, das anhand des Diagramms erstellt werden kann. Der weitere Parameter, coordinate_2, berechnet die zweite Koordinate.
norm	Die optionale Normalisierungsmethode wird vor dem KMeans-Clustering auf Datensätze angewendet. Mögliche Werte: 0 oder „none“ für keine Normalisierung. 1 oder „zscore“ für z-score-Normalisierung 2 oder „minmax“ für min-max-Normalisierung Wenn kein Parameter bereitgestellt wird oder wenn der bereitgestellte Parameter falsch ist, wird keine Normalisierung angewendet. z-score normalisiert Daten gestützt auf Funktionsmittel und Standardabweichung. z-score stellt nicht sicher, dass jede Funktion die gleiche Skala hat, ist aber im Fall von Ausreißern eine besser geeignete Option als min-max. Min-max-Normalisierung sorgt dafür, dass die Funktionen die gleiche Skala haben, indem jeder Mindest- und Höchstwert erfasst und jeder Datenpunkt neu berechnet wird.

Beispiel: Diagrammformel
In diesem Beispiel erstellen wir ein Punktdiagramm anhand des Datensatzes *Iris* und verwenden dann KMeans, um die Daten nach Formel farblich zu kennzeichnen.

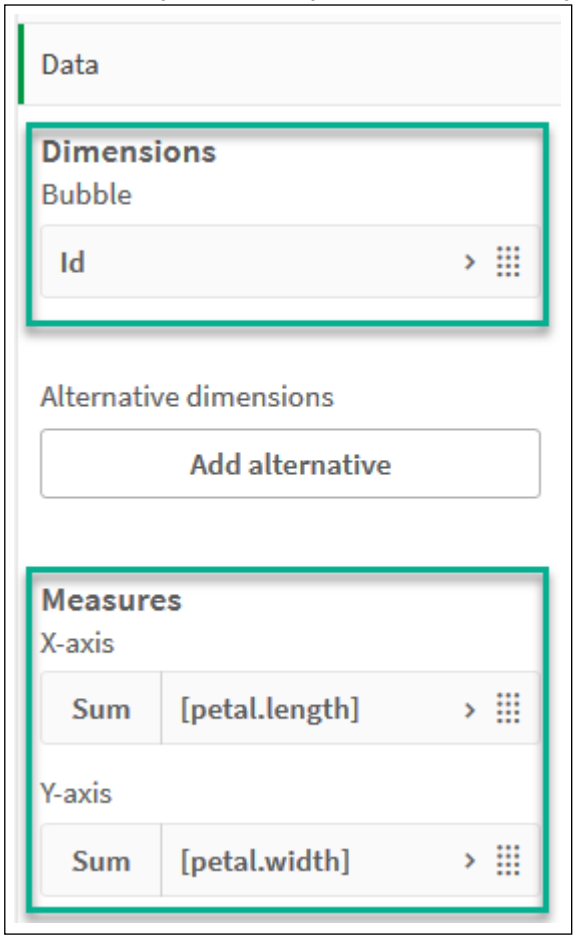
Daneben erstellen wir eine Variable für das Argument *num_clusters* und verwenden dann ein Variableneingabefeld, um die Anzahl der Cluster zu ändern.

Der Datensatz *Iris* ist in verschiedenen Formaten öffentlich verfügbar. Die Daten wurden als Inline-Tabelle bereitgestellt, die mit dem Dateneditor in Qlik Sense geladen werden kann. Beachten Sie, dass für dieses Beispiel der Datentabelle eine Spalte *ID* hinzugefügt wurde.

Nach dem Laden von Daten in Qlik Sense gehen wir wie folgt vor:

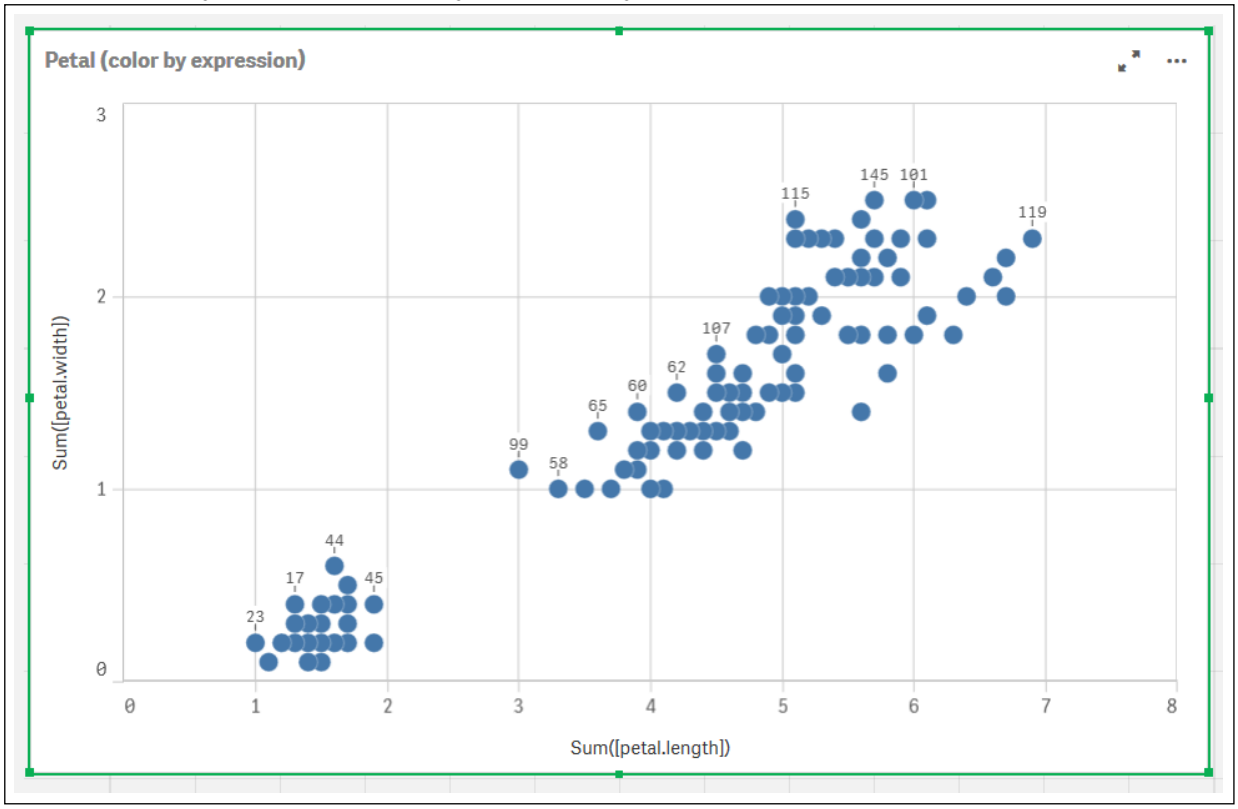
1. Ziehen Sie ein **Punktdiagramm** auf ein neues Arbeitsblatt. Geben Sie dem Diagramm den Namen *Blütenblatt (Farbige Kennzeichnung nach Formel)*.
2. Erstellen Sie eine Variable, um die Anzahl der Cluster anzugeben. Geben Sie für die Variable **Name** *KmeansPetalClusters* ein. Geben Sie für die Variable **Definition** *=2* ein.
3. Konfigurieren Sie **Daten** für das Diagramm:
    i. Wählen Sie unter **Dimensionen** die Option *ID* für das Feld für **Blase**. Geben Sie als Bezeichnung die Cluster-ID ein.
    ii. Wählen Sie unter **Kennzahlen** die Option *Sum([petal.length])* als Formel für **X-Achse**.
    iii. Wählen Sie unter **Kennzahlen** die Option *Sum([petal.width])* als Formel für **Y-Achse**.

    Dateneinstellungen für das Diagramm Blütenblatt (Farbige Kennzeichnung nach Formel)



Die Datenpunkte werden im Diagramm aufgetragen.

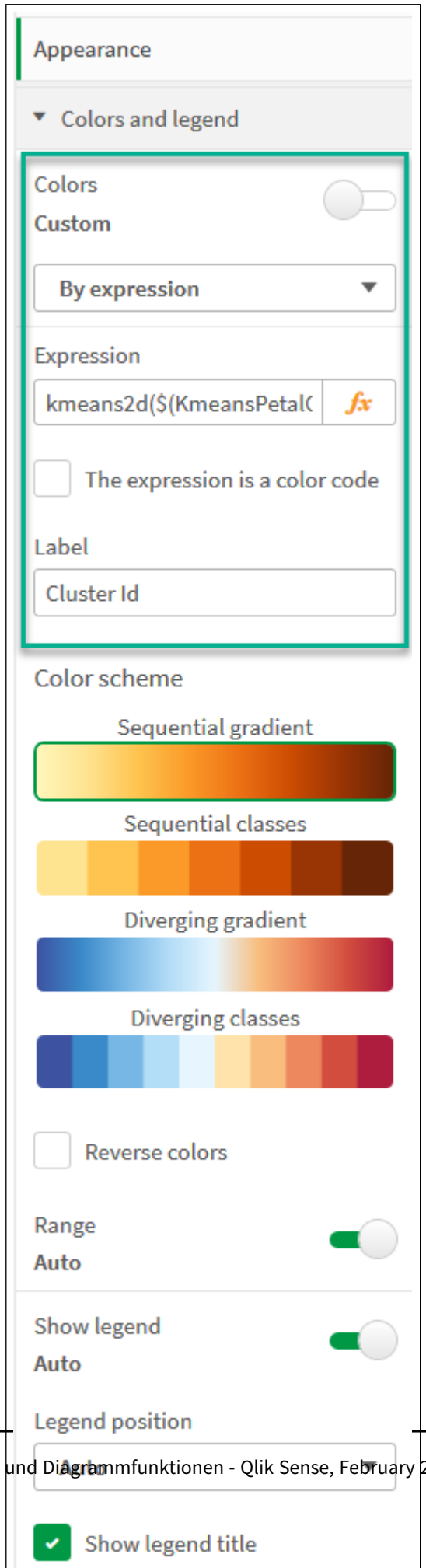
Datenpunkte im Diagramm Blütenblatt (Farbige Kennzeichnung nach Formel)



4. Konfigurieren Sie die **Darstellung** für das Diagramm:
    i. Wählen Sie unter **Farben und Legenden** die Option **Benutzerdefiniert** für **Farben**.
    ii. Wählen Sie, die Farben des Diagramms **Nach Formel** festzulegen.
    iii. Geben Sie für **Formel** Folgendes ein: *kmeans2d($(KmeansPetalClusters), Sum([petal.length]), Sum([petal.width]))*
    Beachten Sie, dass *KmeansPetalClusters* die Variable ist, die wir auf *2* festlegen.
    Geben Sie alternativ Folgendes ein: *kmeans2d(2, Sum([petal.length]), Sum([petal.width]))*
    iv. Deaktivieren Sie das Kontrollkästchen für **Die Formel ist ein Farbcode**.

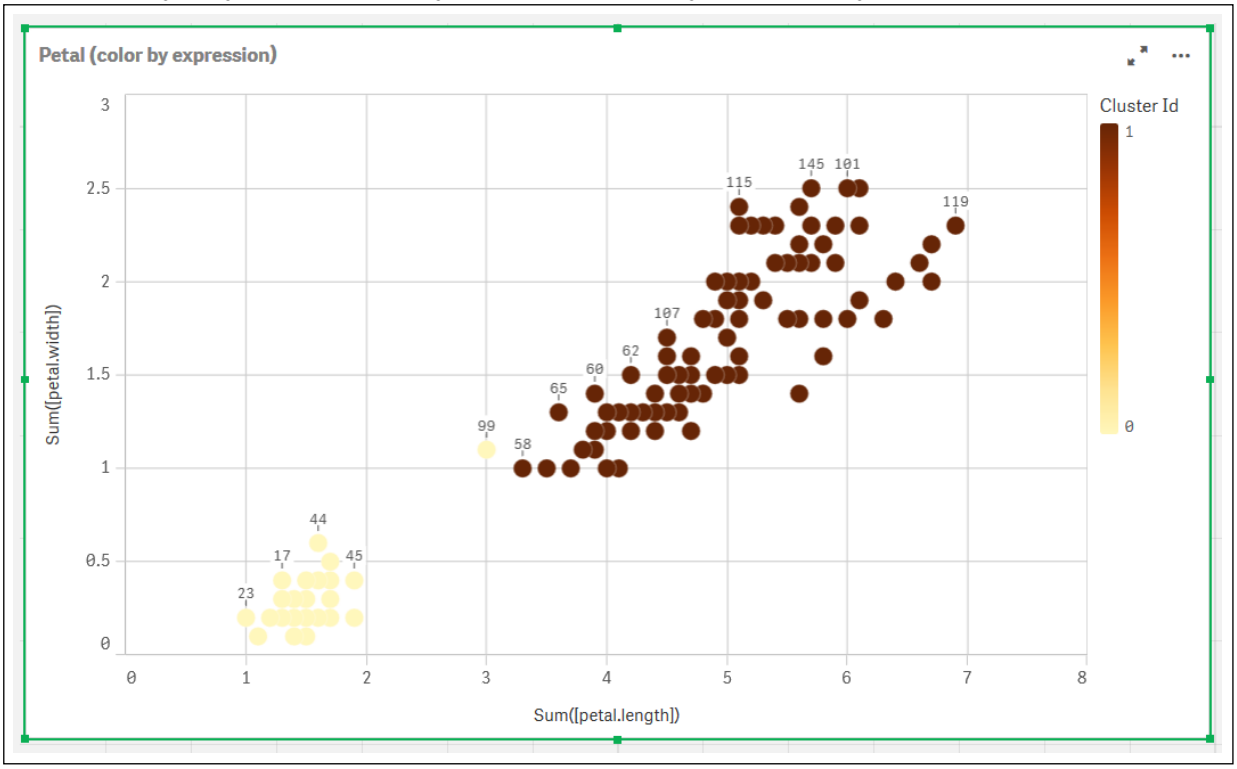
v. Geben Sie Folgendes für **Bezeichnung** ein: *Cluster-ID*

Darstellungseinstellungen für das Diagramm Blütenblatt (Farbige Kennzeichnung nach Formel)


Appearance
Colors and legend
Colors
Custom
By expression
Expression
kmeans2d($(KmeansPetal(
The expression is a color code
Label
Cluster Id
Color scheme
Sequential gradient
Sequential classes
Diverging gradient
Diverging classes
Reverse colors
Range
Auto
Show legend
Auto
Legend position
Auto
Show legend title

Die beiden Cluster im Diagramm erhalten ihre Farbe entsprechend der KMeans-Formel.

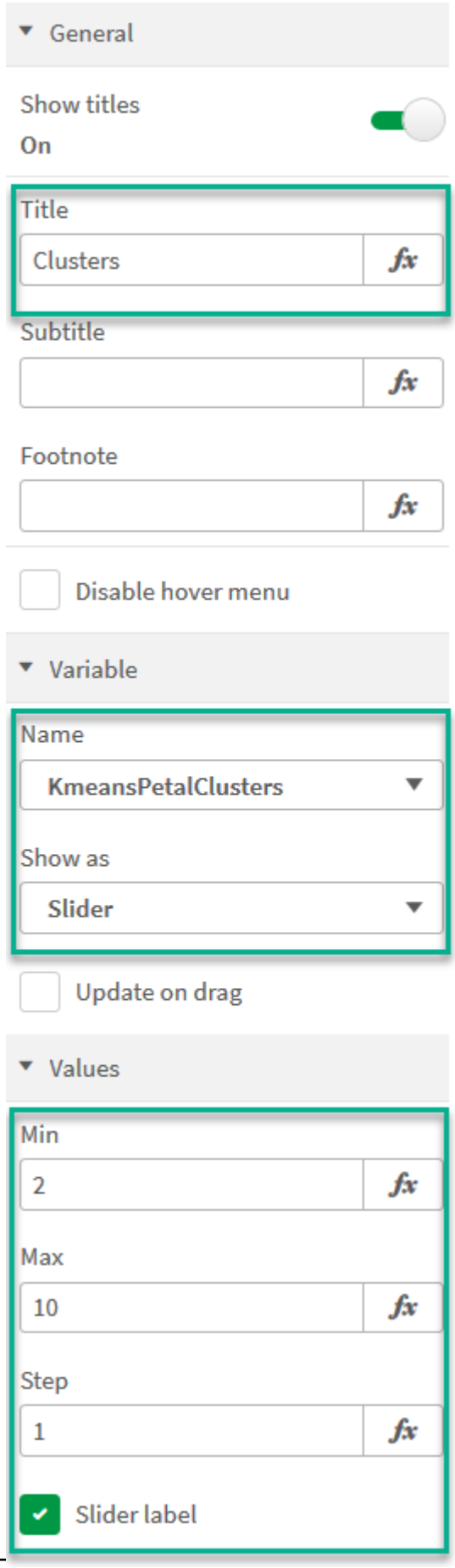
Cluster mit Farbgebung nach Formel im Diagramm Blütenblatt (Farbige Kennzeichnung nach Formel)



5. Fügen Sie ein **Variableneingabenfeld** für die Anzahl der Cluster hinzu.
    i. Wählen Sie unter **Benutzerdefinierte Objekte** im **Extras**-Fenster die Option **Qlik Dashboard Bundle**. Wenn kein Zugriff auf das Dashboard Bundle besteht, kann die Anzahl der Cluster dennoch mithilfe der erstellten Variable oder direkt als ganze Zahl in der Formel geändert werden.
    ii. Ziehen Sie ein **Variableneingabenfeld** auf das Arbeitsblatt.
    iii. Klicken Sie unter **Darstellung** auf **Allgemein**.
    iv. Geben Sie Folgendes als **Titel** ein: *Cluster*
    v. Klicken Sie auf **Variable**.
    vi. Wählen Sie die folgende Variable für **Name**: *KmeansPetalClusters*.
    vii. Wählen Sie **Schieberegler** für **Anzeigen als**.

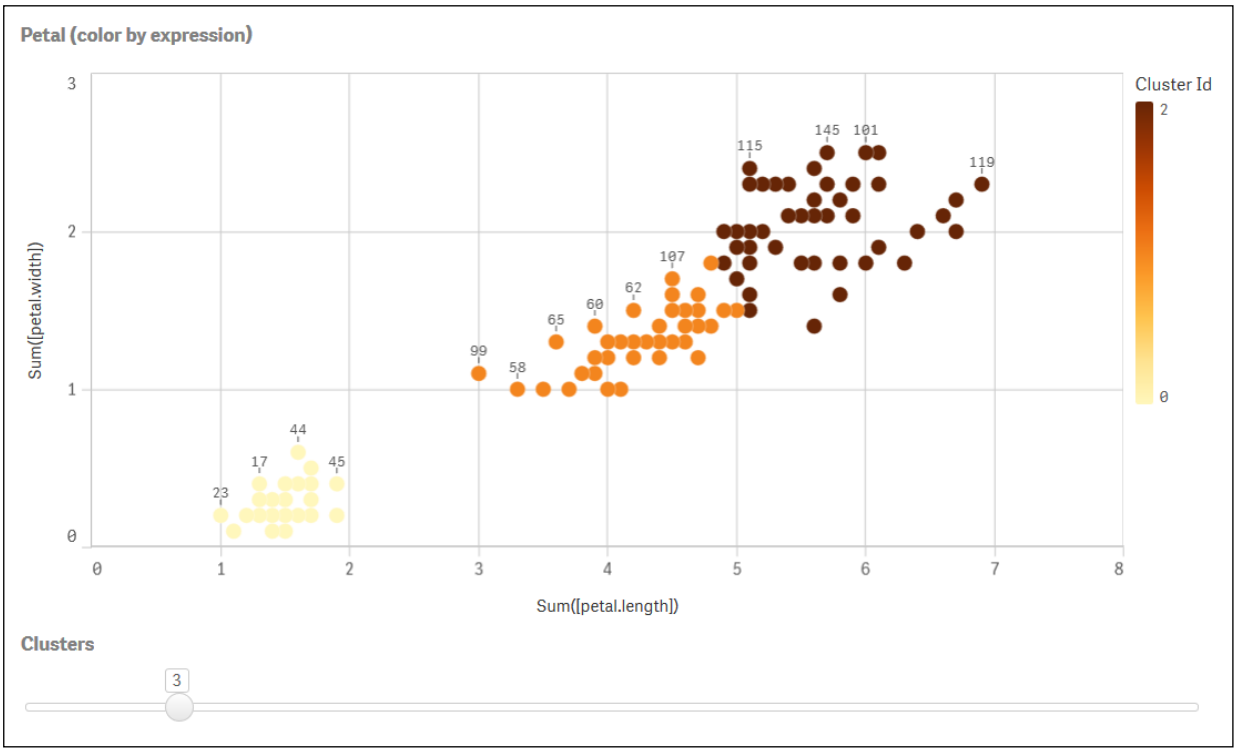
viii. Wählen Sie **Werte** und konfigurieren Sie die Einstellungen wie erforderlich.

Darstellung für das Variableneingabefeld Cluster


General
Show titles
On
Title
Clusters
Subtitle
Footnote
Disable hover menu
Variable
Name
KmeansPetalClusters
Show as
Slider
Update on drag
Values
Min
2
Max
10
Step
1
Slider label

Nach Abschluss der Bearbeitung kann die Anzahl der Cluster anhand des Schiebereglers im Variableneingabefeld *Cluster* geändert werden.

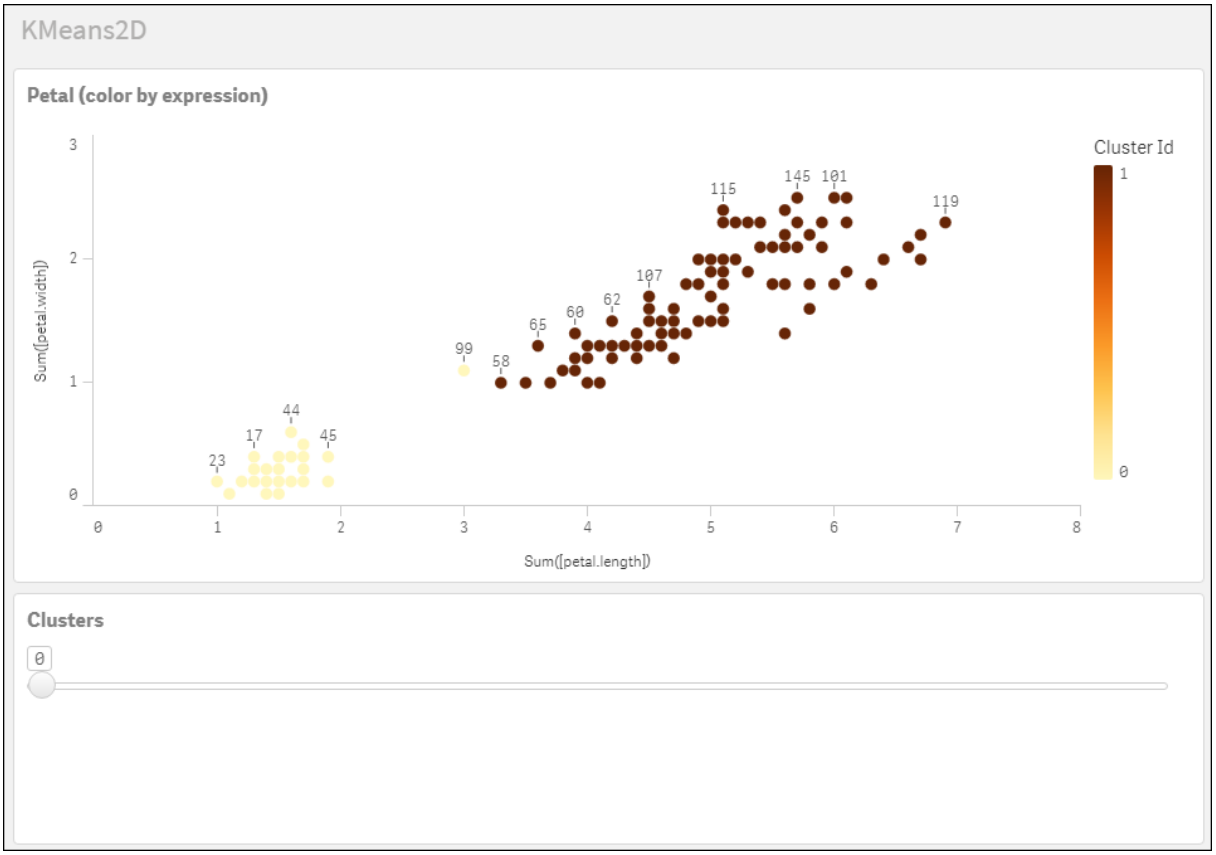
Cluster mit Farbgebung nach Formel im Diagramm Blütenblatt (Farbige Kennzeichnung nach Formel)



Automatisches Clustering

KMeans-Funktionen unterstützen automatisches Clustering mit einer Methode, die als Tiefendifferenz bezeichnet wird. Wenn ein Benutzer 0 für die Anzahl der Cluster festlegt, wird eine optimale Anzahl Cluster für diesen Datensatz bestimmt. Beachten Sie, dass eine Ganzzahl für die Anzahl der Cluster (*k*) nicht explizit zurückgegeben, sondern im Rahmen des KMeans-Algorithmus berechnet wird. Wenn beispielsweise 0 in der Funktion für den Wert von *KmeansPetalClusters* oder über ein Variableneingabefeld festgelegt wird, werden Clusterzuweisungen automatisch für den Datensatz gestützt auf eine optimale Anzahl Cluster berechnet.

Die KMeans-Tiefendifferenzmethode bestimmt die optimale Anzahl an Clustern, wenn (k) auf 0 festgelegt wird.



Iris-Datensatz: Inline-Ladevorgang für Dateneditor in Qlik Sense

```
IrisData:
Load * Inline [
sepal.length, sepal.width, petal.length, petal.width, variety, id
5.1, 3.5, 1.4, 0.2, Setosa, 1
4.9, 3, 1.4, 0.2, Setosa, 2
4.7, 3.2, 1.3, 0.2, Setosa, 3
4.6, 3.1, 1.5, 0.2, Setosa, 4
5, 3.6, 1.4, 0.2, Setosa, 5
5.4, 3.9, 1.7, 0.4, Setosa, 6
4.6, 3.4, 1.4, 0.3, Setosa, 7
5, 3.4, 1.5, 0.2, Setosa, 8
4.4, 2.9, 1.4, 0.2, Setosa, 9
4.9, 3.1, 1.5, 0.1, Setosa, 10
5.4, 3.7, 1.5, 0.2, Setosa, 11
4.8, 3.4, 1.6, 0.2, Setosa, 12
4.8, 3, 1.4, 0.1, Setosa, 13
4.3, 3, 1.1, 0.1, Setosa, 14
5.8, 4, 1.2, 0.2, Setosa, 15
5.7, 4.4, 1.5, 0.4, Setosa, 16
5.4, 3.9, 1.3, 0.4, Setosa, 17
5.1, 3.5, 1.4, 0.3, Setosa, 18
5.7, 3.8, 1.7, 0.3, Setosa, 19
5.1, 3.8, 1.5, 0.3, Setosa, 20
5.4, 3.4, 1.7, 0.2, Setosa, 21
```

```
5.1, 3.7, 1.5, 0.4, Setosa, 22
4.6, 3.6, 1, 0.2, Setosa, 23
5.1, 3.3, 1.7, 0.5, Setosa, 24
4.8, 3.4, 1.9, 0.2, Setosa, 25
5, 3, 1.6, 0.2, Setosa, 26
5, 3.4, 1.6, 0.4, Setosa, 27
5.2, 3.5, 1.5, 0.2, Setosa, 28
5.2, 3.4, 1.4, 0.2, Setosa, 29
4.7, 3.2, 1.6, 0.2, Setosa, 30
4.8, 3.1, 1.6, 0.2, Setosa, 31
5.4, 3.4, 1.5, 0.4, Setosa, 32
5.2, 4.1, 1.5, 0.1, Setosa, 33
5.5, 4.2, 1.4, 0.2, Setosa, 34
4.9, 3.1, 1.5, 0.1, Setosa, 35
5, 3.2, 1.2, 0.2, Setosa, 36
5.5, 3.5, 1.3, 0.2, Setosa, 37
4.9, 3.1, 1.5, 0.1, Setosa, 38
4.4, 3, 1.3, 0.2, Setosa, 39
5.1, 3.4, 1.5, 0.2, Setosa, 40
5, 3.5, 1.3, 0.3, Setosa, 41
4.5, 2.3, 1.3, 0.3, Setosa, 42
4.4, 3.2, 1.3, 0.2, Setosa, 43
5, 3.5, 1.6, 0.6, Setosa, 44
5.1, 3.8, 1.9, 0.4, Setosa, 45
4.8, 3, 1.4, 0.3, Setosa, 46
5.1, 3.8, 1.6, 0.2, Setosa, 47
4.6, 3.2, 1.4, 0.2, Setosa, 48
5.3, 3.7, 1.5, 0.2, Setosa, 49
5, 3.3, 1.4, 0.2, Setosa, 50
7, 3.2, 4.7, 1.4, Versicolor, 51
6.4, 3.2, 4.5, 1.5, Versicolor, 52
6.9, 3.1, 4.9, 1.5, Versicolor, 53
5.5, 2.3, 4, 1.3, Versicolor, 54
6.5, 2.8, 4.6, 1.5, Versicolor, 55
5.7, 2.8, 4.5, 1.3, Versicolor, 56
6.3, 3.3, 4.7, 1.6, Versicolor, 57
4.9, 2.4, 3.3, 1, Versicolor, 58
6.6, 2.9, 4.6, 1.3, Versicolor, 59
5.2, 2.7, 3.9, 1.4, Versicolor, 60
5, 2, 3.5, 1, Versicolor, 61
5.9, 3, 4.2, 1.5, Versicolor, 62
6, 2.2, 4, 1, Versicolor, 63
6.1, 2.9, 4.7, 1.4, Versicolor, 64
5.6, 2.9, 3.6, 1.3, Versicolor, 65
6.7, 3.1, 4.4, 1.4, Versicolor, 66
5.6, 3, 4.5, 1.5, Versicolor, 67
5.8, 2.7, 4.1, 1, Versicolor, 68
6.2, 2.2, 4.5, 1.5, Versicolor, 69
5.6, 2.5, 3.9, 1.1, Versicolor, 70
5.9, 3.2, 4.8, 1.8, Versicolor, 71
6.1, 2.8, 4, 1.3, Versicolor, 72
6.3, 2.5, 4.9, 1.5, Versicolor, 73
6.1, 2.8, 4.7, 1.2, Versicolor, 74
6.4, 2.9, 4.3, 1.3, Versicolor, 75
6.6, 3, 4.4, 1.4, Versicolor, 76
```

```
6.8, 2.8, 4.8, 1.4, Versicolor, 77
6.7, 3, 5, 1.7, Versicolor, 78
6, 2.9, 4.5, 1.5, Versicolor, 79
5.7, 2.6, 3.5, 1, Versicolor, 80
5.5, 2.4, 3.8, 1.1, Versicolor, 81
5.5, 2.4, 3.7, 1, Versicolor, 82
5.8, 2.7, 3.9, 1.2, Versicolor, 83
6, 2.7, 5.1, 1.6, Versicolor, 84
5.4, 3, 4.5, 1.5, Versicolor, 85
6, 3.4, 4.5, 1.6, Versicolor, 86
6.7, 3.1, 4.7, 1.5, Versicolor, 87
6.3, 2.3, 4.4, 1.3, Versicolor, 88
5.6, 3, 4.1, 1.3, Versicolor, 89
5.5, 2.5, 4, 1.3, Versicolor, 90
5.5, 2.6, 4.4, 1.2, Versicolor, 91
6.1, 3, 4.6, 1.4, Versicolor, 92
5.8, 2.6, 4, 1.2, Versicolor, 93
5, 2.3, 3.3, 1, Versicolor, 94
5.6, 2.7, 4.2, 1.3, Versicolor, 95
5.7, 3, 4.2, 1.2, Versicolor, 96
5.7, 2.9, 4.2, 1.3, Versicolor, 97
6.2, 2.9, 4.3, 1.3, Versicolor, 98
5.1, 2.5, 3, 1.1, Versicolor, 99
5.7, 2.8, 4.1, 1.3, Versicolor, 100
6.3, 3.3, 6, 2.5, Virginica, 101
5.8, 2.7, 5.1, 1.9, Virginica, 102
7.1, 3, 5.9, 2.1, Virginica, 103
6.3, 2.9, 5.6, 1.8, Virginica, 104
6.5, 3, 5.8, 2.2, Virginica, 105
7.6, 3, 6.6, 2.1, Virginica, 106
4.9, 2.5, 4.5, 1.7, Virginica, 107
7.3, 2.9, 6.3, 1.8, Virginica, 108
6.7, 2.5, 5.8, 1.8, Virginica, 109
7.2, 3.6, 6.1, 2.5, Virginica, 110
6.5, 3.2, 5.1, 2, Virginica, 111
6.4, 2.7, 5.3, 1.9, Virginica, 112
6.8, 3, 5.5, 2.1, Virginica, 113
5.7, 2.5, 5, 2, Virginica, 114
5.8, 2.8, 5.1, 2.4, Virginica, 115
6.4, 3.2, 5.3, 2.3, Virginica, 116
6.5, 3, 5.5, 1.8, Virginica, 117
7.7, 3.8, 6.7, 2.2, Virginica, 118
7.7, 2.6, 6.9, 2.3, Virginica, 119
6, 2.2, 5, 1.5, Virginica, 120
6.9, 3.2, 5.7, 2.3, Virginica, 121
5.6, 2.8, 4.9, 2, Virginica, 122
7.7, 2.8, 6.7, 2, Virginica, 123
6.3, 2.7, 4.9, 1.8, Virginica, 124
6.7, 3.3, 5.7, 2.1, Virginica, 125
7.2, 3.2, 6, 1.8, Virginica, 126
6.2, 2.8, 4.8, 1.8, Virginica, 127
6.1, 3, 4.9, 1.8, Virginica, 128
6.4, 2.8, 5.6, 2.1, Virginica, 129
7.2, 3, 5.8, 1.6, Virginica, 130
7.4, 2.8, 6.1, 1.9, Virginica, 131
```

```
7.9, 3.8, 6.4, 2, Virginica, 132
6.4, 2.8, 5.6, 2.2, Virginica, 133
6.3, 2.8, 5.1, 1.5, Virginica, 134
6.1, 2.6, 5.6, 1.4, Virginica, 135
7.7, 3, 6.1, 2.3, Virginica, 136
6.3, 3.4, 5.6, 2.4, Virginica, 137
6.4, 3.1, 5.5, 1.8, Virginica, 138
6, 3, 4.8, 1.8, Virginica, 139
6.9, 3.1, 5.4, 2.1, Virginica, 140
6.7, 3.1, 5.6, 2.4, Virginica, 141
6.9, 3.1, 5.1, 2.3, Virginica, 142
5.8, 2.7, 5.1, 1.9, Virginica, 143
6.8, 3.2, 5.9, 2.3, Virginica, 144
6.7, 3.3, 5.7, 2.5, Virginica, 145
6.7, 3, 5.2, 2.3, Virginica, 146
6.3, 2.5, 5, 1.9, Virginica, 147
6.5, 3, 5.2, 2, Virginica, 148
6.2, 3.4, 5.4, 2.3, Virginica, 149
5.9, 3, 5.1, 1.8, Virginica, 150
];
```

KMeansND - Diagrammfunktion

KMeansND() wertet die Zeilen des Diagramms aus, indem K-means-Clustering angewandt wird. Für jede Diagrammzeile wird die Cluster-ID des Clusters angezeigt, dem dieser Datenpunkt zugewiesen wurde. Die vom Clustering-Algorithmus verwendeten Spalten werden von den Parametern coordinate_1, coordinate_2 usw. bis zu n Spalten festgelegt. Es handelt sich bei allen um Aggregierungen. Die Anzahl der erstellten Cluster wird durch den Parameter num_clusters bestimmt.

KMeansND gibt einen Wert pro Datenpunkt zurück. Der zurückgegebene Wert ist ein dualer Wert und ein Ganzzahlwert, der dem Cluster entspricht, dem der jeweilige Datenpunkt zugewiesen wurde.

Syntax:

```
KMeansND(num_clusters, num_iter, coordinate_1, coordinate_2 [,coordinate_3 [,
...]])
```

Rückgabe Datentyp: dual

Argumente:

Argumente

Argument	Beschreibung
num_clusters	Ganze Zahl, die die Anzahl der Cluster angibt
num_iter	Anzahl der Iterationen des Clustering mit neu initialisierten Cluster-Centern
coordinate_1	Die Aggregierung, die die erste Koordinate berechnet, in der Regel die x-Achse eines Punktdiagramms, das anhand des Diagramms erstellt werden kann. Mit den weiteren Parametern werden die zweite, dritte, vierte Koordinate usw. berechnet.

Beispiel: Diagrammformel
In diesem Beispiel erstellen wir ein Punktdiagramm anhand des Datensatzes *Iris* und verwenden dann KMeans, um die Daten nach Formel farblich zu kennzeichnen.

Daneben erstellen wir eine Variable für das Argument *num_clusters* und verwenden dann ein Variableneingabefeld, um die Anzahl der Cluster zu ändern.

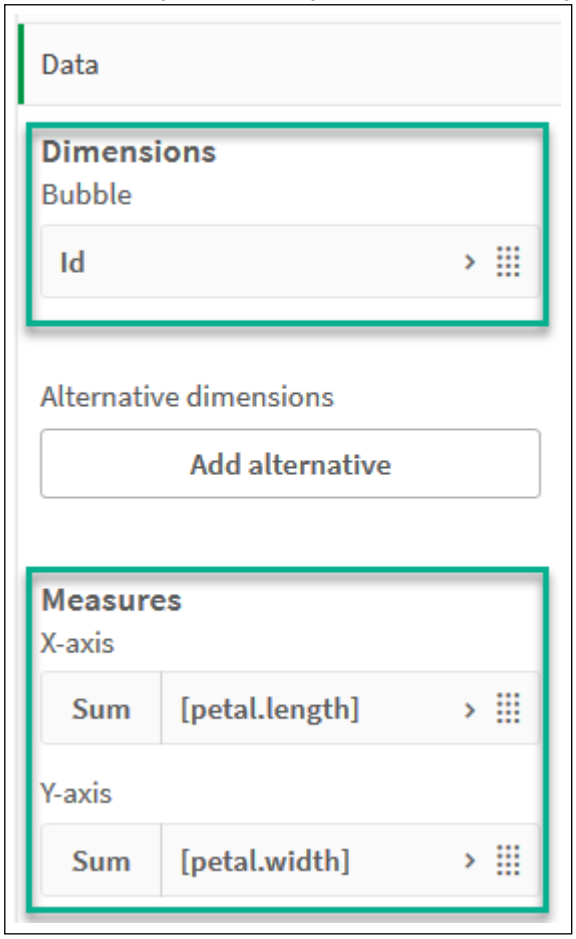
Zudem erstellen wir eine Variable für das Argument *num_iter* und verwenden dann ein zweites Variableneingabefeld, um die Anzahl der Iterationen zu ändern.

Der Datensatz *Iris* ist in verschiedenen Formaten öffentlich verfügbar. Die Daten wurden als Inline-Tabelle bereitgestellt, die mit dem Dateneditor in Qlik Sense geladen werden kann. Beachten Sie, dass für dieses Beispiel der Datentabelle eine Spalte *ID* hinzugefügt wurde.

Nach dem Laden von Daten in Qlik Sense gehen wir wie folgt vor:

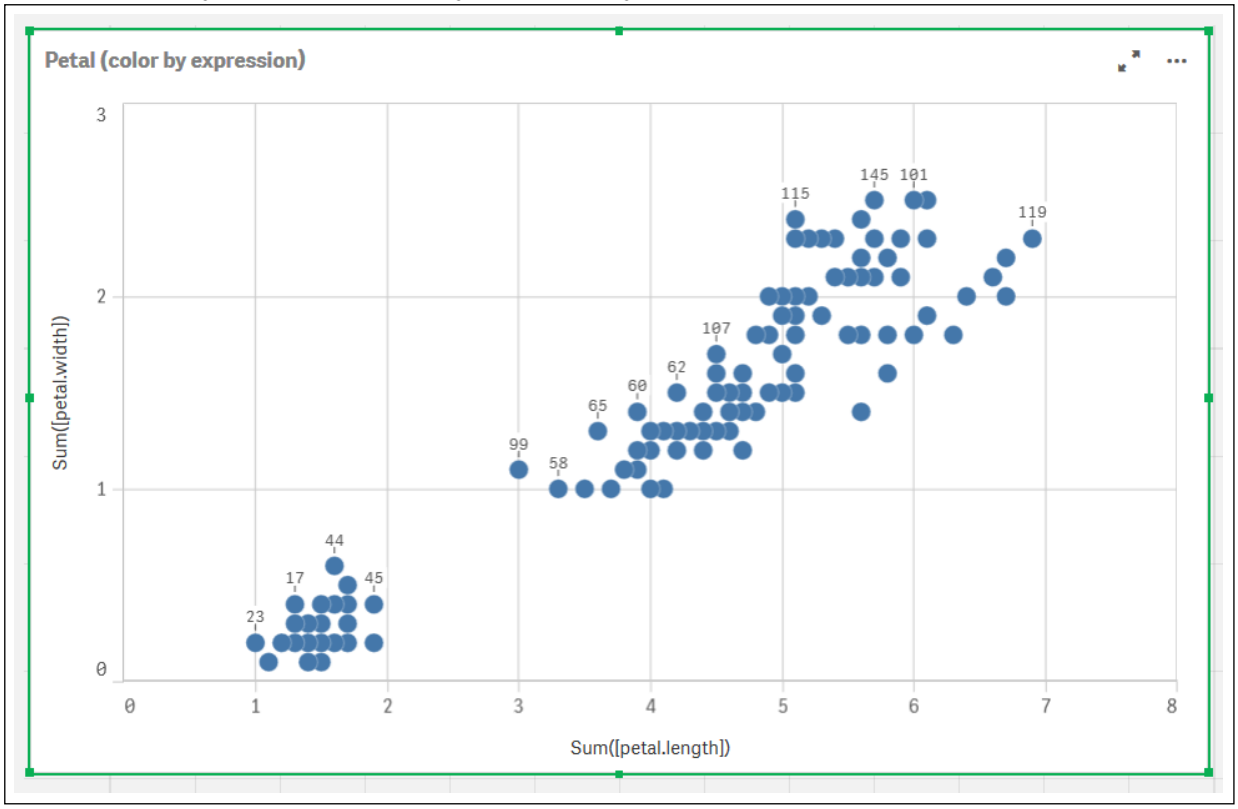
1. Ziehen Sie ein **Punktdiagramm** auf ein neues Arbeitsblatt. Geben Sie dem Diagramm den Namen *Blütenblatt (Farbige Kennzeichnung nach Formel)*.
2. Erstellen Sie eine Variable, um die Anzahl der Cluster anzugeben. Geben Sie für die Variable **Name** *KmeansPetalClusters* ein. Geben Sie für die Variable **Definition** *=2* ein.
3. Erstellen Sie eine Variable, um die Anzahl der Iterationen anzugeben. Geben Sie für die Variable **Name** *KmeansNumberIterations* ein. Geben Sie für die Variable **Definition** *=1* ein.
4. Konfigurieren Sie **Daten** für das Diagramm:
    i. Wählen Sie unter **Dimensionen** die Option *ID* für das Feld für **Blase**. Geben Sie als Bezeichnung die Cluster-ID ein.
    ii. Wählen Sie unter **Kennzahlen** die Option *Sum([petal.length])* als Formel für **X-Achse**.
    iii. Wählen Sie unter **Kennzahlen** die Option *Sum([petal.width])* als Formel für **Y-Achse**.

Dateneinstellungen für das Diagramm Blütenblatt (Farbige Kennzeichnung nach Formel)



Die Datenpunkte werden im Diagramm aufgetragen.

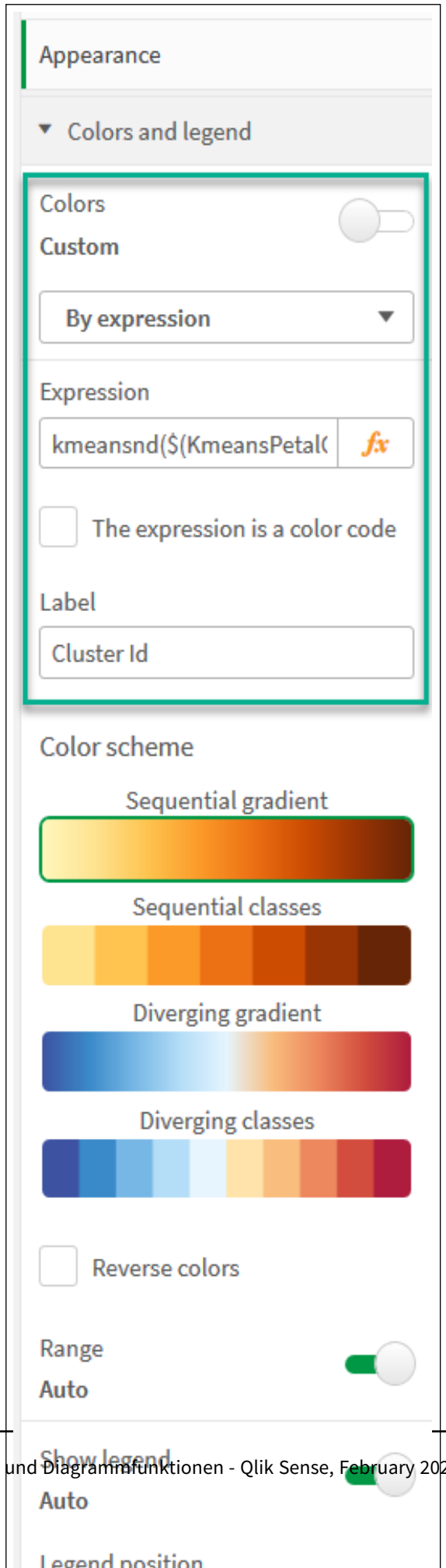
Datenpunkte im Diagramm Blütenblatt (Farbige Kennzeichnung nach Formel)



5. Konfigurieren Sie die **Darstellung** für das Diagramm:
   i. Wählen Sie unter **Farben und Legenden** die Option **Benutzerdefiniert** für **Farben**.
   ii. Wählen Sie, die Farben des Diagramms **Nach Formel** festzulegen.
   iii. Geben Sie für **Formel** Folgendes ein: *kmeansnd ($(KmeansPetalClusters),$(KmeansNumberIterations), Sum([petal.length]), Sum ([petal.width]),Sum([sepal.length]), Sum([sepal.width]))*
   Beachten Sie, dass *KmeansPetalClusters* die Variable ist, die wir auf *2* festlegen.
   KmeansNumberIterations ist die Variable, die wir auf *1* festlegen.
   Geben Sie alternativ Folgendes ein: *kmeansnd(2, 2, Sum([petal.length]), Sum([petal.width]),Sum ([sepal.length]), Sum([sepal.width]))*
   iv. Deaktivieren Sie das Kontrollkästchen für **Die Formel ist ein Farbcode**.

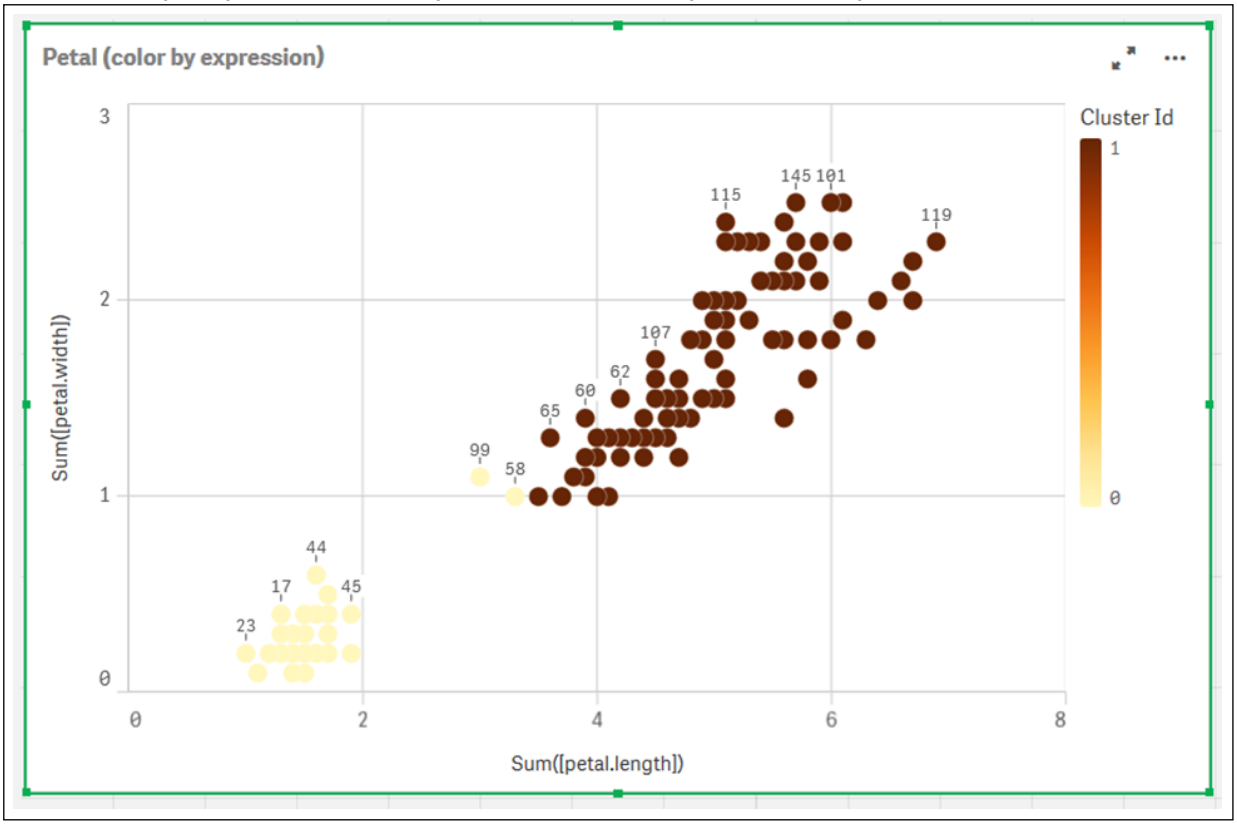
v. Geben Sie Folgendes für **Bezeichnung** ein: *Cluster-ID*

Darstellungseinstellungen für das Diagramm Blütenblatt (Farbige Kennzeichnung nach Formel)


Appearance
Colors and legend
Colors
Custom
By expression
Expression
kmeansnd($(KmeansPetal(
The expression is a color code
Label
Cluster Id
Color scheme
Sequential gradient
Sequential classes
Diverging gradient
Diverging classes
Reverse colors
Range
Auto
Show legend
Auto
Legend position

Die beiden Cluster im Diagramm erhalten ihre Farbe entsprechend der KMeans-Formel.

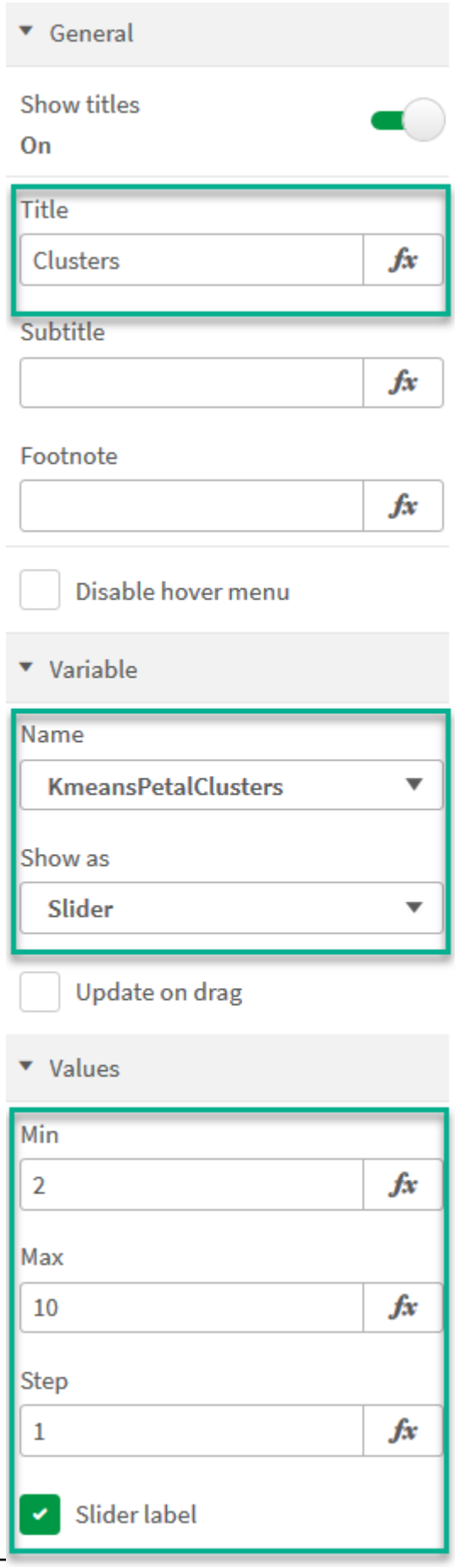
Cluster mit Farbgebung nach Formel im Diagramm Blütenblatt (Farbige Kennzeichnung nach Formel)



6. Fügen Sie ein **Variableneingabenfeld** für die Anzahl der Cluster hinzu.
    i. Wählen Sie unter **Benutzerdefinierte Objekte** im **Extras**-Fenster die Option **Qlik Dashboard Bundle**. Wenn kein Zugriff auf das Dashboard Bundle besteht, kann die Anzahl der Cluster dennoch mithilfe der erstellten Variable oder direkt als ganze Zahl in der Formel geändert werden.
    ii. Ziehen Sie ein **Variableneingabenfeld** auf das Arbeitsblatt.
    iii. Klicken Sie unter **Darstellung** auf **Allgemein**.
    iv. Geben Sie Folgendes als **Titel** ein: *Cluster*
    v. Klicken Sie auf **Variable**.
    vi. Wählen Sie die folgende Variable für **Name**: *KmeansPetalClusters*.
    vii. Wählen Sie **Schieberegler** für **Anzeigen als**.

viii. Wählen Sie **Werte** und konfigurieren Sie die Einstellungen wie erforderlich.

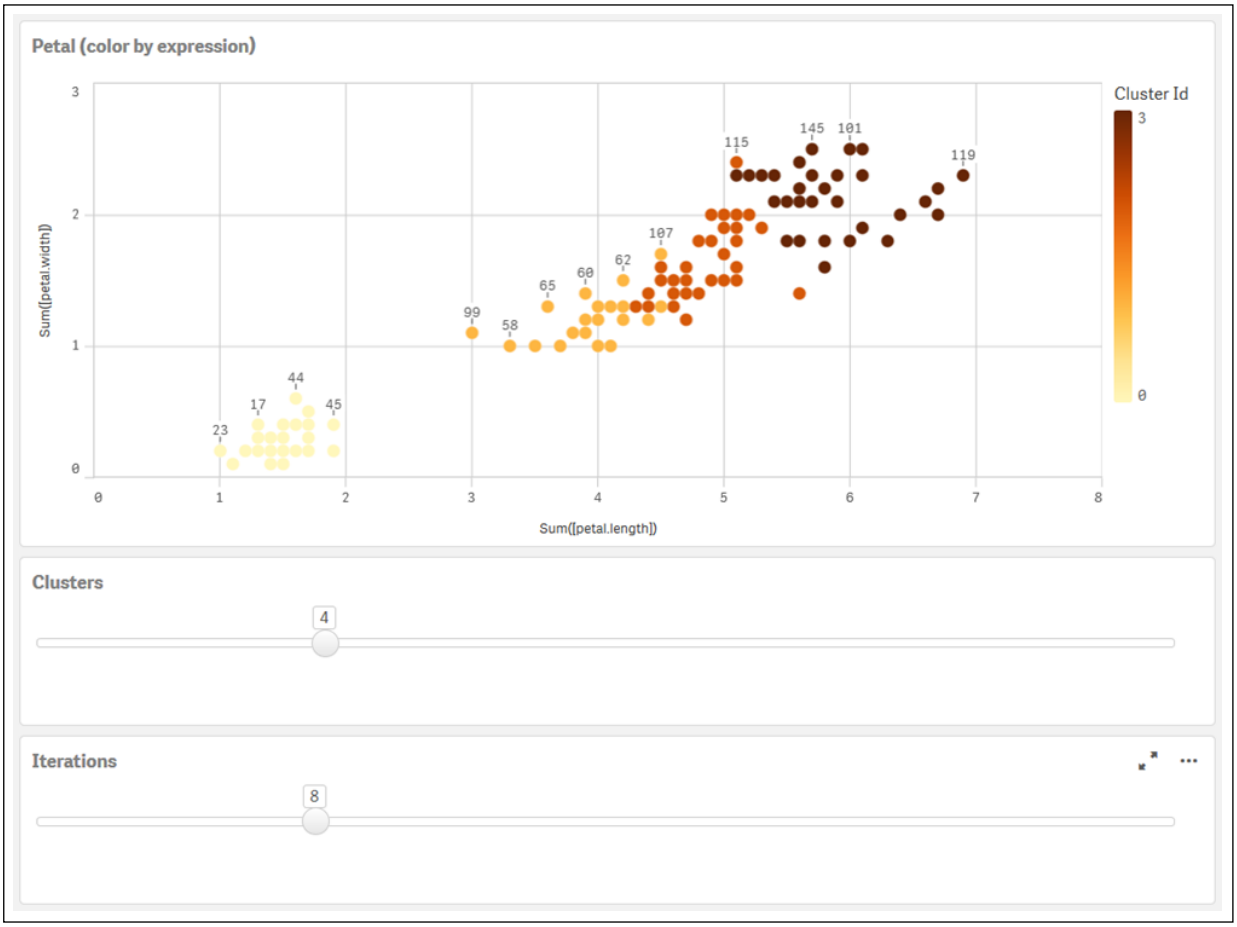
Darstellung für das Variableneingabefeld Cluster


General
Show titles
On
Title
Clusters
Subtitle
Footnote
Disable hover menu
Variable
Name
KmeansPetalClusters
Show as
Slider
Update on drag
Values
Min
2
Max
10
Step
1
Slider label

7. Fügen Sie ein **Variableneingabenfeld** für die Anzahl der Iterationen hinzu.
    i. Ziehen Sie ein **Variableneingabenfeld** auf das Arbeitsblatt.
    ii. Wählen Sie unter **Darstellung** die Option **Allgemein**.
    iii. Geben Sie Folgendes als **Titel** ein: *Iterationen*
    iv. Wählen Sie unter **Darstellung** die Option **Variable**.
    v. Wählen Sie die folgende Variable unter **Name**: *KmeansNumberIterations*.
    vi. Konfigurieren Sie die zusätzlichen Einstellungen nach Bedarf.

    Jetzt kann die Anzahl der Cluster und Iterationen anhand des Schiebereglers in den Variableneingabefeldern geändert werden.

    Cluster mit Farbgebung nach Formel im Diagramm Blütenblatt (Farbige Kennzeichnung nach Formel)

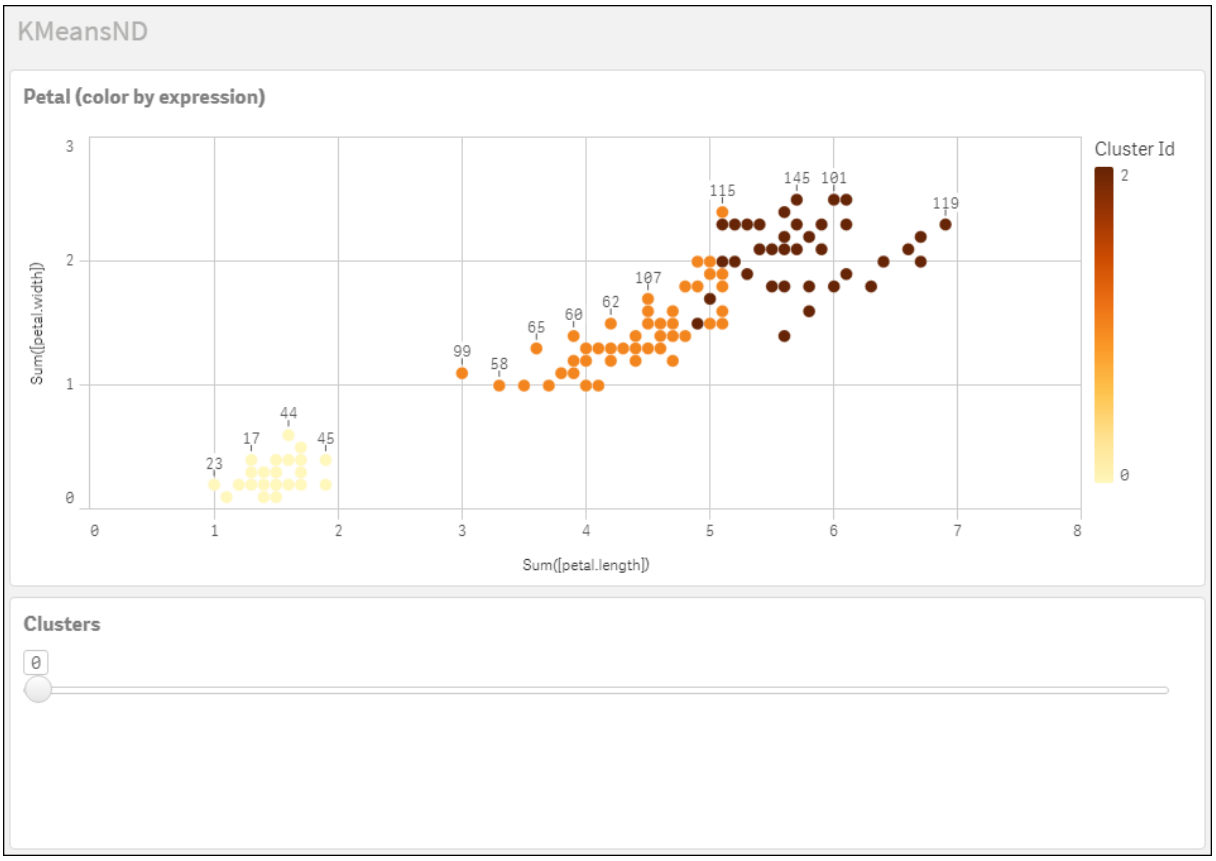


Automatisches Clustering

KMeans-Funktionen unterstützen automatisches Clustering mit einer Methode, die als Tiefendifferenz bezeichnet wird. Wenn ein Benutzer 0 für die Anzahl der Cluster festlegt, wird eine optimale Anzahl Cluster für diesen Datensatz bestimmt. Beachten Sie, dass eine Ganzzahl für die Anzahl der Cluster (*k*) nicht explizit zurückgegeben, sondern im Rahmen des KMeans-Algorithmus berechnet wird. Wenn beispielsweise 0 in der Funktion für den Wert von *KmeansPetalClusters* oder über ein Variableneingabefeld festgelegt wird, werden

Clusterzuweisungen automatisch für den Datensatz gestützt auf eine optimale Anzahl Cluster berechnet. Wenn im Iris-Datensatz 0 als Anzahl der Cluster ausgewählt wird, bestimmt der Algorithmus (mit automatischem Clustering) eine optimale Anzahl Cluster (3) für diesen Datensatz.

Die KMeans-Tiefendifferenzmethode bestimmt die optimale Anzahl an Clustern, wenn (k) auf 0 festgelegt wird.



Iris-Datensatz: Inline-Ladevorgang für Dateneditor in Qlik Sense

```
IrisData:
Load * Inline [
sepal.length, sepal.width, petal.length, petal.width, variety, id
5.1, 3.5, 1.4, 0.2, Setosa, 1
4.9, 3, 1.4, 0.2, Setosa, 2
4.7, 3.2, 1.3, 0.2, Setosa, 3
4.6, 3.1, 1.5, 0.2, Setosa, 4
5, 3.6, 1.4, 0.2, Setosa, 5
5.4, 3.9, 1.7, 0.4, Setosa, 6
4.6, 3.4, 1.4, 0.3, Setosa, 7
5, 3.4, 1.5, 0.2, Setosa, 8
4.4, 2.9, 1.4, 0.2, Setosa, 9
4.9, 3.1, 1.5, 0.1, Setosa, 10
5.4, 3.7, 1.5, 0.2, Setosa, 11
4.8, 3.4, 1.6, 0.2, Setosa, 12
4.8, 3, 1.4, 0.1, Setosa, 13
4.3, 3, 1.1, 0.1, Setosa, 14
5.8, 4, 1.2, 0.2, Setosa, 15
5.7, 4.4, 1.5, 0.4, Setosa, 16
```

```
5.4, 3.9, 1.3, 0.4, Setosa, 17
5.1, 3.5, 1.4, 0.3, Setosa, 18
5.7, 3.8, 1.7, 0.3, Setosa, 19
5.1, 3.8, 1.5, 0.3, Setosa, 20
5.4, 3.4, 1.7, 0.2, Setosa, 21
5.1, 3.7, 1.5, 0.4, Setosa, 22
4.6, 3.6, 1, 0.2, Setosa, 23
5.1, 3.3, 1.7, 0.5, Setosa, 24
4.8, 3.4, 1.9, 0.2, Setosa, 25
5, 3, 1.6, 0.2, Setosa, 26
5, 3.4, 1.6, 0.4, Setosa, 27
5.2, 3.5, 1.5, 0.2, Setosa, 28
5.2, 3.4, 1.4, 0.2, Setosa, 29
4.7, 3.2, 1.6, 0.2, Setosa, 30
4.8, 3.1, 1.6, 0.2, Setosa, 31
5.4, 3.4, 1.5, 0.4, Setosa, 32
5.2, 4.1, 1.5, 0.1, Setosa, 33
5.5, 4.2, 1.4, 0.2, Setosa, 34
4.9, 3.1, 1.5, 0.1, Setosa, 35
5, 3.2, 1.2, 0.2, Setosa, 36
5.5, 3.5, 1.3, 0.2, Setosa, 37
4.9, 3.1, 1.5, 0.1, Setosa, 38
4.4, 3, 1.3, 0.2, Setosa, 39
5.1, 3.4, 1.5, 0.2, Setosa, 40
5, 3.5, 1.3, 0.3, Setosa, 41
4.5, 2.3, 1.3, 0.3, Setosa, 42
4.4, 3.2, 1.3, 0.2, Setosa, 43
5, 3.5, 1.6, 0.6, Setosa, 44
5.1, 3.8, 1.9, 0.4, Setosa, 45
4.8, 3, 1.4, 0.3, Setosa, 46
5.1, 3.8, 1.6, 0.2, Setosa, 47
4.6, 3.2, 1.4, 0.2, Setosa, 48
5.3, 3.7, 1.5, 0.2, Setosa, 49
5, 3.3, 1.4, 0.2, Setosa, 50
7, 3.2, 4.7, 1.4, Versicolor, 51
6.4, 3.2, 4.5, 1.5, Versicolor, 52
6.9, 3.1, 4.9, 1.5, Versicolor, 53
5.5, 2.3, 4, 1.3, Versicolor, 54
6.5, 2.8, 4.6, 1.5, Versicolor, 55
5.7, 2.8, 4.5, 1.3, Versicolor, 56
6.3, 3.3, 4.7, 1.6, Versicolor, 57
4.9, 2.4, 3.3, 1, Versicolor, 58
6.6, 2.9, 4.6, 1.3, Versicolor, 59
5.2, 2.7, 3.9, 1.4, Versicolor, 60
5, 2, 3.5, 1, Versicolor, 61
5.9, 3, 4.2, 1.5, Versicolor, 62
6, 2.2, 4, 1, Versicolor, 63
6.1, 2.9, 4.7, 1.4, Versicolor, 64
5.6, 2.9, 3.6, 1.3, Versicolor, 65
6.7, 3.1, 4.4, 1.4, Versicolor, 66
5.6, 3, 4.5, 1.5, Versicolor, 67
5.8, 2.7, 4.1, 1, Versicolor, 68
6.2, 2.2, 4.5, 1.5, Versicolor, 69
5.6, 2.5, 3.9, 1.1, Versicolor, 70
5.9, 3.2, 4.8, 1.8, Versicolor, 71
```

```
6.1, 2.8, 4, 1.3, Versicolor, 72
6.3, 2.5, 4.9, 1.5, Versicolor, 73
6.1, 2.8, 4.7, 1.2, Versicolor, 74
6.4, 2.9, 4.3, 1.3, Versicolor, 75
6.6, 3, 4.4, 1.4, Versicolor, 76
6.8, 2.8, 4.8, 1.4, Versicolor, 77
6.7, 3, 5, 1.7, Versicolor, 78
6, 2.9, 4.5, 1.5, Versicolor, 79
5.7, 2.6, 3.5, 1, Versicolor, 80
5.5, 2.4, 3.8, 1.1, Versicolor, 81
5.5, 2.4, 3.7, 1, Versicolor, 82
5.8, 2.7, 3.9, 1.2, Versicolor, 83
6, 2.7, 5.1, 1.6, Versicolor, 84
5.4, 3, 4.5, 1.5, Versicolor, 85
6, 3.4, 4.5, 1.6, Versicolor, 86
6.7, 3.1, 4.7, 1.5, Versicolor, 87
6.3, 2.3, 4.4, 1.3, Versicolor, 88
5.6, 3, 4.1, 1.3, Versicolor, 89
5.5, 2.5, 4, 1.3, Versicolor, 90
5.5, 2.6, 4.4, 1.2, Versicolor, 91
6.1, 3, 4.6, 1.4, Versicolor, 92
5.8, 2.6, 4, 1.2, Versicolor, 93
5, 2.3, 3.3, 1, Versicolor, 94
5.6, 2.7, 4.2, 1.3, Versicolor, 95
5.7, 3, 4.2, 1.2, Versicolor, 96
5.7, 2.9, 4.2, 1.3, Versicolor, 97
6.2, 2.9, 4.3, 1.3, Versicolor, 98
5.1, 2.5, 3, 1.1, Versicolor, 99
5.7, 2.8, 4.1, 1.3, Versicolor, 100
6.3, 3.3, 6, 2.5, Virginica, 101
5.8, 2.7, 5.1, 1.9, Virginica, 102
7.1, 3, 5.9, 2.1, Virginica, 103
6.3, 2.9, 5.6, 1.8, Virginica, 104
6.5, 3, 5.8, 2.2, Virginica, 105
7.6, 3, 6.6, 2.1, Virginica, 106
4.9, 2.5, 4.5, 1.7, Virginica, 107
7.3, 2.9, 6.3, 1.8, Virginica, 108
6.7, 2.5, 5.8, 1.8, Virginica, 109
7.2, 3.6, 6.1, 2.5, Virginica, 110
6.5, 3.2, 5.1, 2, Virginica, 111
6.4, 2.7, 5.3, 1.9, Virginica, 112
6.8, 3, 5.5, 2.1, Virginica, 113
5.7, 2.5, 5, 2, Virginica, 114
5.8, 2.8, 5.1, 2.4, Virginica, 115
6.4, 3.2, 5.3, 2.3, Virginica, 116
6.5, 3, 5.5, 1.8, Virginica, 117
7.7, 3.8, 6.7, 2.2, Virginica, 118
7.7, 2.6, 6.9, 2.3, Virginica, 119
6, 2.2, 5, 1.5, Virginica, 120
6.9, 3.2, 5.7, 2.3, Virginica, 121
5.6, 2.8, 4.9, 2, Virginica, 122
7.7, 2.8, 6.7, 2, Virginica, 123
6.3, 2.7, 4.9, 1.8, Virginica, 124
6.7, 3.3, 5.7, 2.1, Virginica, 125
7.2, 3.2, 6, 1.8, Virginica, 126
```

```
6.2, 2.8, 4.8, 1.8, Virginica, 127
6.1, 3, 4.9, 1.8, Virginica, 128
6.4, 2.8, 5.6, 2.1, Virginica, 129
7.2, 3, 5.8, 1.6, Virginica, 130
7.4, 2.8, 6.1, 1.9, Virginica, 131
7.9, 3.8, 6.4, 2, Virginica, 132
6.4, 2.8, 5.6, 2.2, Virginica, 133
6.3, 2.8, 5.1, 1.5, Virginica, 134
6.1, 2.6, 5.6, 1.4, Virginica, 135
7.7, 3, 6.1, 2.3, Virginica, 136
6.3, 3.4, 5.6, 2.4, Virginica, 137
6.4, 3.1, 5.5, 1.8, Virginica, 138
6, 3, 4.8, 1.8, Virginica, 139
6.9, 3.1, 5.4, 2.1, Virginica, 140
6.7, 3.1, 5.6, 2.4, Virginica, 141
6.9, 3.1, 5.1, 2.3, Virginica, 142
5.8, 2.7, 5.1, 1.9, Virginica, 143
6.8, 3.2, 5.9, 2.3, Virginica, 144
6.7, 3.3, 5.7, 2.5, Virginica, 145
6.7, 3, 5.2, 2.3, Virginica, 146
6.3, 2.5, 5, 1.9, Virginica, 147
6.5, 3, 5.2, 2, Virginica, 148
6.2, 3.4, 5.4, 2.3, Virginica, 149
5.9, 3, 5.1, 1.8, Virginica, 150
];
```

KMeansCentroid2D - Diagrammfunktion

KMeansCentroid2D() wertet die Zeilen des Diagramms aus, indem K-means-Clustering angewandt wird. Für jede Diagrammzeile wird die gewünschte Koordinate des Clusters angezeigt, dem dieser Datenpunkt zugewiesen wurde. Die vom Clustering-Algorithmus verwendeten Spalten werden von den Parametern coordinate_1 bzw. coordinate_2 festgelegt. Es handelt sich bei beiden um Aggregierungen. Die Anzahl der erstellten Cluster wird durch den Parameter num_clusters bestimmt. Daten können optional mit dem Normparameter normalisiert werden.

KMeansCentroid2D gibt einen Wert pro Datenpunkt zurück. Der zurückgegebene Wert ist ein dualer Wert und ist eine der Koordinaten der Position des Cluster-Zentrums, dem der jeweilige Datenpunkt zugewiesen wurde.

Syntax:

```
KMeansCentroid2D(num_clusters, coordinate_no, coordinate_1, coordinate_2 [, norm])
```

Rückgabe Datentyp: dual

Argumente:

Argumente

Argument	Beschreibung
num_clusters	Ganze Zahl, die die Anzahl der Cluster angibt

Argument	Beschreibung
coordinate_no	Die gewünschte Koordinatennummer der Zentroiden (z. B. entsprechend der x-, y- oder z-Achse).
coordinate_1	Die Aggregierung, die die erste Koordinate berechnet, in der Regel die x-Achse des Punktdiagramms, das anhand des Diagramms erstellt werden kann. Der weitere Parameter, coordinate_2, berechnet die zweite Koordinate.
norm	Die optionale Normalisierungsmethode wird vor dem KMeans-Clustering auf Datensätze angewendet. Mögliche Werte: 0 oder „none“ für keine Normalisierung. 1 oder „zscore“ für z-score-Normalisierung 2 oder „minmax“ für min-max-Normalisierung Wenn kein Parameter bereitgestellt wird oder wenn der bereitgestellte Parameter falsch ist, wird keine Normalisierung angewendet. z-score normalisiert Daten gestützt auf Funktionsmittel und Standardabweichung. z-score stellt nicht sicher, dass jede Funktion die gleiche Skala hat, ist aber im Fall von Ausreißern eine besser geeignete Option als min-max. Min-max-Normalisierung sorgt dafür, dass die Funktionen die gleiche Skala haben, indem jeder Mindest- und Höchstwert erfasst und jeder Datenpunkt neu berechnet wird.

Automatisches Clustering

KMeans-Funktionen unterstützen automatisches Clustering mit einer Methode, die als Tiefendifferenz bezeichnet wird. Wenn ein Benutzer 0 für die Anzahl der Cluster festlegt, wird eine optimale Anzahl Cluster für diesen Datensatz bestimmt. Beachten Sie, dass eine Ganzzahl für die Anzahl der Cluster (*k*) nicht explizit zurückgegeben, sondern im Rahmen des KMeans-Algorithmus berechnet wird. Wenn beispielsweise 0 in der Funktion für den Wert von *KmeansPetalClusters* oder über ein Variableneingabefeld festgelegt wird, werden Clusterzuweisungen automatisch für den Datensatz gestützt auf eine optimale Anzahl Cluster berechnet.

KMeansCentroidND - Diagrammfunktion

KMeansCentroidND() wertet die Zeilen des Diagramms aus, indem K-means-Clustering angewandt wird. Für jede Diagrammzeile wird die gewünschte Koordinate des Clusters angezeigt, dem dieser Datenpunkt zugewiesen wurde. Die vom Clustering-Algorithmus verwendeten Spalten werden von den Parametern coordinate_1, coordinate_2 usw. bis zu n Spalten festgelegt. Es handelt sich bei allen um Aggregierungen. Die Anzahl der erstellten Cluster wird durch den Parameter num_clusters bestimmt.

KMeansCentroidND gibt einen Wert pro Zeile zurück. Der zurückgegebene Wert ist ein dualer Wert und ist eine der Koordinaten der Position des Cluster-Zentrums, dem der jeweilige Datenpunkt zugewiesen wurde.

Syntax:

```
KMeansCentroidND((num_clusters, num_iter, coordinate_no, coordinate_1,
coordinate_2 [,coordinate_3 [, ...]])
```

Rückgabe Datentyp: dual

Argumente:

Argumente

Argument	Beschreibung
num_clusters	Ganze Zahl, die die Anzahl der Cluster angibt
num_iter	Anzahl der Iterationen des Clustering mit neu initialisierten Cluster-Centern
coordinate_no	Die gewünschte Koordinatennummer der Zentroiden (z. B. entsprechend der x-, y- oder z-Achse).
coordinate_1	Die Aggregierung, die die erste Koordinate berechnet, in der Regel die x-Achse eines Punktdiagramms, das anhand des Diagramms erstellt werden kann. Mit den weiteren Parametern werden die zweite, dritte, vierte Koordinate usw. berechnet.

Automatisches Clustering

KMeans-Funktionen unterstützen automatisches Clustering mit einer Methode, die als Tiefendifferenz bezeichnet wird. Wenn ein Benutzer 0 für die Anzahl der Cluster festlegt, wird eine optimale Anzahl Cluster für diesen Datensatz bestimmt. Beachten Sie, dass eine Ganzzahl für die Anzahl der Cluster (*k*) nicht explizit zurückgegeben, sondern im Rahmen des KMeans-Algorithmus berechnet wird. Wenn beispielsweise 0 in der Funktion für den Wert von *KmeansPetalClusters* oder über ein Variableneingabefeld festgelegt wird, werden Clusterzuweisungen automatisch für den Datensatz gestützt auf eine optimale Anzahl Cluster berechnet.

STL_Trend - Diagrammfunktion

STL_Trend ist eine Zeitreihenzerlegungsfunktion. Zusammen mit **STL_Seasonal** und **STL_Residual** wird diese Funktion verwendet, um eine Zeitreihe in Saison-, Trend- und Restkomponenten zu zerlegen. Im Kontext des STL-Algorithmus wird Zeitreihenzerlegung verwendet, um sowohl Saisonmuster als auch einen allgemeinen Trend aus einer Eingabemetrik und anderen Parametern zu ermitteln. Die Funktion **STL_Trend** identifiziert einen allgemeinen, von Saisonmustern oder -zyklen unabhängigen Trend in den Zeitreihendaten.

Die drei STL-Funktionen hängen über eine einfache Summe mit der Eingabemetrik zusammen:

STL_Trend + **STL_Seasonal** + **STL_Residual** = Eingabemetrik

STL (Saison-Trend-Zerlegung mittels LOESS) nutzt Datenglättungstechniken und ermöglicht es den Benutzern anhand von Eingabeparametern, die Periodizität der durchgeführten Berechnungen anzupassen. Diese Periodizität bestimmt, wie die Zeitdimension der Eingabemetrik (eine Kennzahl) in der Analyse segmentiert wird.

STL_Trend benötigt mindestens eine Eingabemetrik (`target_measure`) und einen Ganzzahlwert für seinen `period_int` und gibt einen Gleitkommawert zurück. Die Eingabemetrik hat die Form einer Aggregierung, die entlang der Zeitdimension variiert. Optional können Sie Werte für `seasonal_smoother` und `trend_smoother` einschließen, um den Glättungsalgorithmus anzupassen.

Sie können mit dieser Funktion arbeiten, indem Sie sie direkt im Formel-Editor für ein Diagramm eingeben, oder indem Sie den Modifikator **Zeitreihenzerlegung** zu einer Kennzahl hinzufügen. Sie wird auch in Insight Advisor als Analysetyp verwendet.

Syntax:

```
STL_Trend(target_measure, period_int [,seasonal_smoother [,trend_smoother]])
```

Rückgabe Datentyp: dual

Argumente

Argument	Beschreibung
target_measure	Die Kennzahl zum Zerlegen in Saison- und Trendkomponenten. Dies kann eine Kennzahl wie Sum(Sales) oder Sum(Passengers) sein, die entlang der Zeitdimension variiert. Dabei muss es sich nicht um einen konstanten Wert handeln.
period_int	Die Periodizität des Datensatzes. Dieser Parameter ist ein Ganzzahlwert, der die Anzahl der diskreten Schritte darstellt, aus denen sich ein Zeitraum bzw. Saisonzyklus des Signals zusammensetzt. Wenn beispielsweise die Zeitreihe in einen Abschnitt für jedes Vierteljahr segmentiert wird, müssen Sie den **period_int** auf einen Wert von 4 festlegen, um die Periodizität als Jahr zu definieren.
seasonal_smoother	Länge des Saisonglätters. Dabei muss es sich um eine ungerade Ganzzahl handeln. Der Saisonglätter verwendet Daten für eine bestimmte Phase in der Saisonvariation über eine Reihe von Zeiträumen. Ein diskreter Schritt der Zeitdimension wird für jeden Zeitraum verwendet. Der Saisonglätter gibt die Anzahl der Zeiträume an, die für die Glättung verwendet werden. Wenn beispielsweise die Zeitdimension nach Monat segmentiert wird und der Zeitraum Jahr (12) ist, wird die Saisonkomponente so berechnet, dass jeder einzelne Monat jeden Jahres anhand von Daten für den gleichen Monat berechnet wird, sowohl in diesem Jahr als auch in den angrenzenden Jahren. Der Wert **seasonal_smoother** ist die Anzahl der Jahre, die zur Glättung verwendet werden.
trend_smoother	Länge des Trendglätters. Dabei muss es sich um eine ungerade Ganzzahl handeln. Der Trendglätter verwendet die gleiche Zeitskala wie der Parameter **period_int**, und sein Wert ist die Anzahl der Körner, die zur Glättung verwendet werden. Wenn z. B. eine Zeitreihe nach Monat segmentiert wird, ist der Trendglätter die Anzahl der Monate, die zum Glätten verwendet werden.

Die Diagrammfunktion **STL_Trend** wird oft in Kombination mit den folgenden Funktionen verwendet:

Verwandte Funktionen

Funktion	Interaktion
STL_Seasonal - Diagrammfunktion (page 1453)	Mit dieser Funktion wird die Saisonkomponente einer Zeitreihe berechnet.
STL_Residual - Diagrammfunktion (page 1455)	Wenn eine Eingabemetrik in eine Saison- und eine Trendkomponente zerlegt wird, passt ein Teil der Kennzahlvariationen in keine der beiden Hauptkomponenten. Die Funktion **STL_ Residual** berechnet diesen Teil der Zerlegung.

Ein Tutorial mit einem vollständigen Beispiel für die Verwendung dieser Funktion finden Sie unter *Tutorial – Zeitreihenzerlegung in Qlik Sense (page 1457)*.

STL_Seasonal - Diagrammfunktion

STL_Seasonal ist eine Zeitreihenzerlegungsfunktion. Zusammen mit **STL_Trend** und **STL_ Residual** wird diese Funktion verwendet, um eine Zeitreihe in Saison-, Trend- und Restkomponenten zu zerlegen. Im Kontext des STL-Algorithmus wird Zeitreihenzerlegung verwendet, um sowohl Saisonmuster als auch einen allgemeinen Trend aus einer Eingabemetrik und anderen Parametern zu ermitteln. Die Funktion **STL_Seasonal** kann ein Saisonmuster in einer Zeitreihe identifizieren und es vom allgemeinen, in den Daten angezeigten Trend trennen.

Die drei STL-Funktionen hängen über eine einfache Summe mit der Eingabemetrik zusammen:

STL_Trend + **STL_Seasonal** + **STL_Residual** = Eingabemetrik

STL (Saison-Trend-Zerlegung mittels LOESS) nutzt Datenglättungstechniken und ermöglicht es den Benutzern anhand von Eingabeparametern, die Periodizität der durchgeführten Berechnungen anzupassen. Diese Periodizität bestimmt, wie die Zeitdimension der Eingabemetrik (eine Kennzahl) in der Analyse segmentiert wird.

STL_Seasonal benötigt mindestens eine Eingabemetrik (`target_measure`) und einen Ganzzahlwert für seinen `period_int` und gibt einen Gleitkommawert zurück. Die Eingabemetrik hat die Form einer Aggregierung, die entlang der Zeitdimension variiert. Optional können Sie Werte für `seasonal_smoother` und `trend_smoother` einschließen, um den Glättungsalgorithmus anzupassen.

Sie können mit dieser Funktion arbeiten, indem Sie sie direkt im Formel-Editor für ein Diagramm eingeben, oder indem Sie den Modifikator **Zeitreihenzerlegung** zu einer Kennzahl hinzufügen. Sie wird auch in Insight Advisor als Analysetyp verwendet.

Syntax:

```
STL_Seasonal(target_measure, period_int [,seasonal_smoother [,trend_
smoother]])
```

Rückgabe Datentyp: dual

Argumente

Argument	Beschreibung
target_ measure	Die Kennzahl zum Zerlegen in Saison- und Trendkomponenten. Dies kann eine Kennzahl wie Sum(Sales) oder Sum(Passengers) sein, die entlang der Zeitdimension variiert. Dabei muss es sich nicht um einen konstanten Wert handeln.
period_int	Die Periodizität des Datensatzes. Dieser Parameter ist ein Ganzzahlwert, der die Anzahl der diskreten Schritte darstellt, aus denen sich ein Zeitraum bzw. Saisonzyklus des Signals zusammensetzt. Wenn beispielsweise die Zeitreihe in einen Abschnitt für jedes Vierteljahr segmentiert wird, müssen Sie den **period_int** auf einen Wert von 4 festlegen, um die Periodizität als Jahr zu definieren.
seasonal_ smoother	Länge des Saisonglätters. Dabei muss es sich um eine ungerade Ganzzahl handeln. Der Saisonglätter verwendet Daten für eine bestimmte Phase in der Saisonvariation über eine Reihe von Zeiträumen. Ein diskreter Schritt der Zeitdimension wird für jeden Zeitraum verwendet. Der Saisonglätter gibt die Anzahl der Zeiträume an, die für die Glättung verwendet werden. Wenn beispielsweise die Zeitdimension nach Monat segmentiert wird und der Zeitraum Jahr (12) ist, wird die Saisonkomponente so berechnet, dass jeder einzelne Monat jeden Jahres anhand von Daten für den gleichen Monat berechnet wird, sowohl in diesem Jahr als auch in den angrenzenden Jahren. Der Wert **seasonal_smoother** ist die Anzahl der Jahre, die zur Glättung verwendet werden.
trend_ smoother	Länge des Trendglätters. Dabei muss es sich um eine ungerade Ganzzahl handeln. Der Trendglätter verwendet die gleiche Zeitskala wie der Parameter **period_int**, und sein Wert ist die Anzahl der Körner, die zur Glättung verwendet werden. Wenn z. B. eine Zeitreihe nach Monat segmentiert wird, ist der Trendglätter die Anzahl der Monate, die zum Glätten verwendet werden.

Die Diagrammfunktion **STL_Seasonal** wird oft in Kombination mit den folgenden Funktionen verwendet:

Verwandte Funktionen

Funktion	Interaktion
STL_Trend - Diagrammfunktion (page 1451)	Mit dieser Funktion wird die Trendkomponente einer Zeitreihe berechnet.
STL_Residual - Diagrammfunktion (page 1455)	Wenn eine Eingabemetrik in eine Saison- und eine Trendkomponente zerlegt wird, passt ein Teil der Kennzahlvariationen in keine der beiden Hauptkomponenten. Die Funktion **STL_ Residual** berechnet diesen Teil der Zerlegung.

Ein Tutorial mit einem vollständigen Beispiel für die Verwendung dieser Funktion finden Sie unter *Tutorial – Zeitreihenzerlegung in Qlik Sense (page 1457)*.

STL_Residual - Diagrammfunktion

STL_Residual ist eine Zeitreihenzerlegungsfunktion. Zusammen mit **STL_Seasonal** und **STL_ Trend** wird diese Funktion verwendet, um eine Zeitreihe in Saison-, Trend- und Restkomponenten zu zerlegen. Im Kontext des STL-Algorithmus wird Zeitreihenzerlegung verwendet, um sowohl Saisonmuster als auch einen allgemeinen Trend aus einer Eingabemetrik und anderen Parametern zu ermitteln. Wenn diese Operation ausgeführt wird, passen manche der Variationen in der Eingabemetrik weder zur Saison- noch zur Trendkomponente und werden als Restkomponente definiert. Die Diagrammfunktion **STL_Residual** erfasst diesen Teil der Berechnung.

Die drei STL-Funktionen hängen über eine einfache Summe mit der Eingabemetrik zusammen:

STL_Trend + **STL_Seasonal** + **STL_Residual** = Eingabemetrik

STL (Saison-Trend-Zerlegung mittels LOESS) nutzt Datenglättungstechniken und ermöglicht es den Benutzern anhand von Eingabeparametern, die Periodizität der durchgeführten Berechnungen anzupassen. Diese Periodizität bestimmt, wie die Zeitdimension der Eingabemetrik (eine Kennzahl) in der Analyse segmentiert wird.

Da die Zeitreihenzerlegung vor allem nach Saisonalität und allgemeinen Variationen in Daten sucht, werden die Informationen im Rest als die am wenigsten bedeutenden der drei Komponenten betrachtet. Eine verzerrte oder periodische Restkomponente kann aber dazu beitragen, Probleme in der Berechnung zu identifizieren, beispielsweise falsche Zeitraumeinstellungen.

STL_Residual benötigt mindestens eine Eingabemetrik (`target_measure`) und einen Ganzzahlwert für seinen `period_int` und gibt einen Gleitkommawert zurück. Die Eingabemetrik hat die Form einer Aggregierung, die entlang der Zeitdimension variiert. Optional können Sie Werte für `seasonal_smoother` und `trend_smoother` einschließen, um den Glättungsalgorithmus anzupassen.

Sie können mit dieser Funktion arbeiten, indem Sie sie direkt im Formel-Editor für ein Diagramm eingeben, oder indem Sie den Modifikator **Zeitreihenzerlegung** zu einer Kennzahl hinzufügen. Sie wird auch in Insight Advisor als Analysetyp verwendet.

Syntax:

```
STL_Residual(target_measure, period_int [,seasonal_smoother [,trend_
smoother]])
```

Rückgabe Datentyp: dual

Argumente

Argument	Beschreibung
target_measure	Die Kennzahl zum Zerlegen in Saison- und Trendkomponenten. Dies kann eine Kennzahl wie Sum(Sales) oder Sum(Passengers) sein, die entlang der Zeitdimension variiert. Dabei muss es sich nicht um einen konstanten Wert handeln.
period_int	Die Periodizität des Datensatzes. Dieser Parameter ist ein Ganzzahlwert, der die Anzahl der diskreten Schritte darstellt, aus denen sich ein Zeitraum bzw. Saisonzyklus des Signals zusammensetzt. Wenn beispielsweise die Zeitreihe in einen Abschnitt für jedes Vierteljahr segmentiert wird, müssen Sie den **period_int** auf einen Wert von 4 festlegen, um die Periodizität als Jahr zu definieren.
seasonal_smoother	Länge des Saisonglätters. Dabei muss es sich um eine ungerade Ganzzahl handeln. Der Saisonglätter verwendet Daten für eine bestimmte Phase in der Saisonvariation über eine Reihe von Zeiträumen. Ein diskreter Schritt der Zeitdimension wird für jeden Zeitraum verwendet. Der Saisonglätter gibt die Anzahl der Zeiträume an, die für die Glättung verwendet werden. Wenn beispielsweise die Zeitdimension nach Monat segmentiert wird und der Zeitraum Jahr (12) ist, wird die Saisonkomponente so berechnet, dass jeder einzelne Monat jeden Jahres anhand von Daten für den gleichen Monat berechnet wird, sowohl in diesem Jahr als auch in den angrenzenden Jahren. Der Wert **seasonal_smoother** ist die Anzahl der Jahre, die zur Glättung verwendet werden.

Argument	Beschreibung
trend_ smoother	Länge des Trendglätters. Dabei muss es sich um eine ungerade Ganzzahl handeln. Der Trendglätter verwendet die gleiche Zeitskala wie der Parameter **period_int**, und sein Wert ist die Anzahl der Körner, die zur Glättung verwendet werden. Wenn z. B. eine Zeitreihe nach Monat segmentiert wird, ist der Trendglätter die Anzahl der Monate, die zum Glätten verwendet werden.

Die Diagrammfunktion **STL_Residual** wird oft in Kombination mit den folgenden Funktionen verwendet:

Verwandte Funktionen

Funktion	Interaktion
STL_Seasonal - Diagrammfunktion (page 1453)	Mit dieser Funktion wird die Saisonkomponente einer Zeitreihe berechnet.
STL_Trend - Diagrammfunktion (page 1451)	Mit dieser Funktion wird die Trendkomponente einer Zeitreihe berechnet.

Ein Tutorial mit einem vollständigen Beispiel für die Verwendung dieser Funktion finden Sie unter *Tutorial – Zeitreihenzerlegung in Qlik Sense (page 1457)*.

Tutorial – Zeitreihenzerlegung in Qlik Sense

Dieses Tutorial zeigt die Verwendung von drei Diagrammfunktionen zum Zerlegen einer Zeitreihe anhand des STL-Algorithmus.

Dieses Tutorial verwendet Zeitreihendaten für die Anzahl der Passagiere einer Fluggesellschaft pro Monat, um die Funktionsweise des STL-Algorithmus zu zeigen. Die Diagrammfunktionen **STL_Trend**, **STL_Seasonal** und **STL_Residual** werden verwendet, um die Visualisierungen zu erstellen. Weitere Informationen über die Zeitreihenzerlegung in Qlik Sense finden Sie unter *Zeitreihenzerlegungs-Funktionen (page 1403)*.

App erstellen

Beginnen Sie, indem Sie eine neue App erstellen und den Datensatz darin importieren.

Laden Sie den folgenden Datensatz herunter:

Tutorial – Zeitreihenzerlegung

Diese Datei enthält Daten bezüglich der Anzahl Passagiere einer Fluggesellschaft pro Monat.

Gehen Sie folgendermaßen vor:

1. Klicken Sie im Hub auf **Neue App erstellen**.
2. Öffnen Sie die App und legen Sie die Datei *Tutorial - Time series decomposition.csv* darin ab.

Vorbereiten und Laden der Daten

Damit Qlik Sense das Feld „YearMonth“ korrekt interpretiert, müssen Sie möglicherweise den Datenmanager verwenden, damit das Feld als Datumsfeld und nicht als Feld mit Stringwerten erkannt wird. In der Regel erfolgt dieser Schritt automatisch, aber in diesem Fall weisen die Datumswerte das eher ungewöhnliche Format *JJJJ-MM* auf.

1. Wählen Sie im Datenmanager die Tabelle aus und klicken Sie auf .
2. Klicken Sie bei ausgewähltem Feld *YearMonth* auf und legen Sie den **Feldtyp** auf **Datum** fest.
3. Geben Sie unter **Eingabeformat** den Wert *JJJJ-MM* ein.
4. Geben Sie unter **Anzeigeformat** den Wert *JJJJ-MM* ein und klicken Sie auf **OK**.
   Das Feld zeigt jetzt das Kalendersymbol an.
5. Klicken Sie auf **Daten laden**.

Jetzt sind Sie bereit, mit der Verwendung der STL-Funktionen zu beginnen, um Ihre Daten visuell darzustellen.

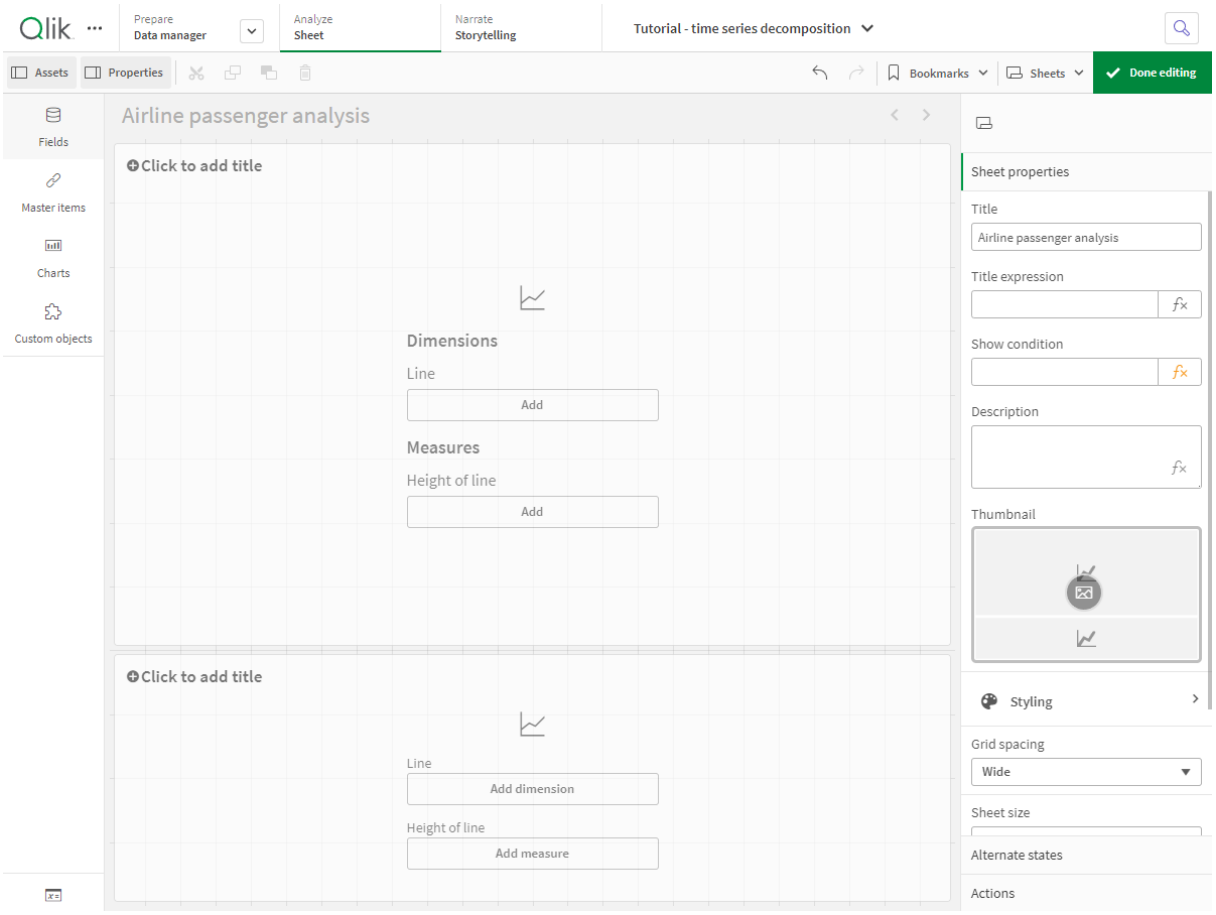
Erstellen der Visualisierungen

Als nächstes erstellen Sie zwei Liniendiagramme, um die Funktionalität der Diagrammfunktionen **STL_Trend**, **STL_Seasonal** und **STL_Residual** zu zeigen.

Öffnen Sie ein neues Arbeitsblatt und geben Sie ihm einen Titel.

Fügen Sie zwei Liniendiagramme zum Arbeitsblatt hinzu. Ändern Sie die Größe und die Position der Diagramme entsprechend der folgenden Abbildung.

Qlik Sense Rasterumriss eines leeren App-Arbeitsblatts



Erstes Liniendiagramm: Trend- und Saisonkomponenten

Gehen Sie folgendermaßen vor:

1. Fügen Sie den Titel *Saison und Trend* zum ersten Liniendiagramm hinzu.
2. Fügen Sie *YearMonth* als Dimension hinzu und nennen Sie sie *Datum*.
3. Fügen Sie die folgende Kennzahl hinzu und nennen Sie sie *Passagiere pro Monat*:
   =Sum(Passengers)
4. Erweitern Sie unter **Daten** die Kennzahl *Passagiere pro Monat* und klicken Sie auf **Trendlinie hinzufügen**.
5. Legen Sie den **Typ** auf **Linear** fest.
   Sie vergleichen diese Trendlinie dann mit der geglätteten Ausgabe der Trendkomponente.
6. Fügen Sie die folgende Kennzahl hinzu, um die Trendkomponente zu plotten, und nennen Sie sie *Trend*:
   =STL_Trend(SUM(Passengers), 12)
7. Fügen Sie die folgende Kennzahl hinzu, um die Saisonkomponente zu plotten, und nennen Sie sie *Saisonal*:
   =STL_Seasonal(SUM(Passengers), 12)

8. Legen Sie unter **Darstellung** > **Präsentation** den Wert für **Scroll-Leiste** auf **Keine** fest.
9. Behalten Sie die Standardfarben bei oder ändern Sie sie wie gewünscht.

Zweites Liniendiagramm: Restkomponente

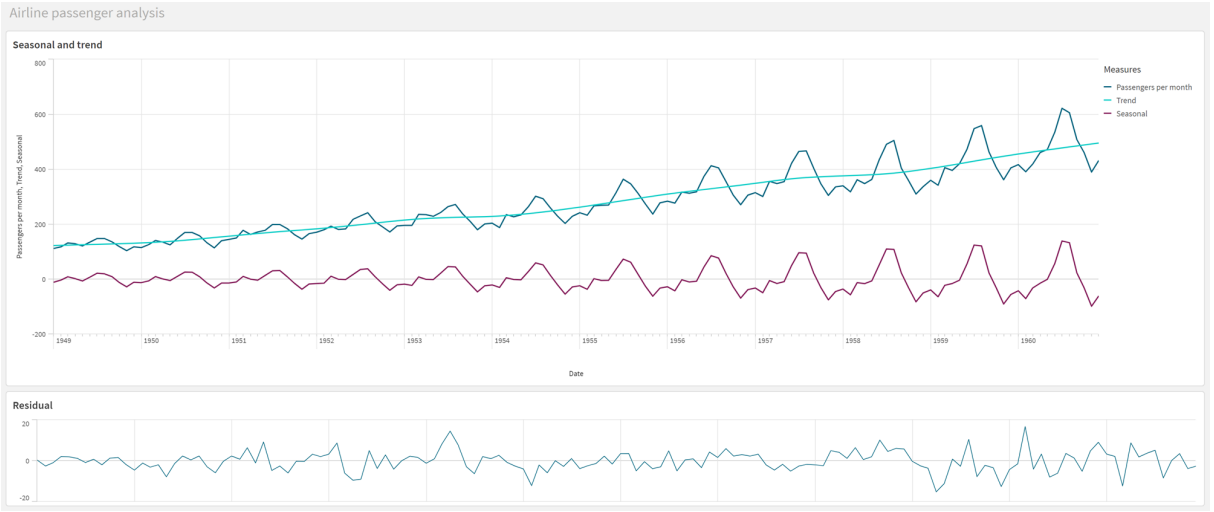
Konfigurieren Sie dann das zweite Liniendiagramm. In dieser Visualisierung wird die Restkomponente der Zeitreihe angezeigt.

Gehen Sie folgendermaßen vor:

1. Ziehen Sie ein Liniendiagramm auf das Arbeitsblatt. Fügen Sie den Titel *Rest* hinzu.
2. Fügen Sie *Date* als Dimension hinzu.
3. Fügen Sie die folgende Kennzahl hinzu und nennen Sie sie *Rest*:
   =STL_Residual(SUM(Passengers), 12)
4. Legen Sie unter **Darstellung** > **Präsentation** den Wert für **Scroll-Leiste** auf **Keine** fest.

Ihr Arbeitsblatt sollte der Abbildung unten gleichen.

Qlik Sense-Arbeitsblatt für Flugpassagier-Analyse



Interpretieren und Erläutern der Daten

Mit den STL-Diagrammfunktionen erhalten wir eine Reihe von Einblicken aus unseren Zeitreihendaten.

Trendkomponente

Die statistischen Informationen in der Trendkomponente sind saisonbereinigt. Dadurch wird es einfacher, allgemeine, sich nicht im Zeitverlauf wiederholende Fluktuationen zu erkennen. Im Gegensatz zur geraden, linearen Trendlinie für *Passagiere pro Monat* erfasst die STL-Trendkomponente Trendänderungen. Sie zeigt einige klare Abweichungen, präsentiert die Information aber dennoch in einer leserlichen Art. Dies konnte mithilfe des Glättungsverhaltens im STL-Algorithmus besser erfasst werden.

Die fallenden Zahlen für Flugpassagiere, die im STL-Trenddiagramm angezeigt werden, können im Rahmen der wirtschaftlichen Auswirkung der Rezessionen in den 1950er Jahren erklärt werden.

Saisonkomponente

Die trendbereinigte Saisonkomponente isolierte wiederkehrende Fluktuationen im Lauf der Zeitreihe und entfernte allgemeine Trendinformationen aus diesem Teil der Analyse. Wir begannen mit einem Datensatz, der aus Jahr-Monat-Aggregierungen bestand. Angesichts dieser Daten wird implizit davon ausgegangen, dass wir die Daten in monatlichen Mengen segmentieren. Indem wir einen Zeitraumwert von 12 definieren, legen wir das Diagramm auf das Modellieren von Saisonmustern im Lauf von Einjahres- bzw. Zwölfmonatszyklen fest.

In den Daten findet sich ein wiederholtes Saisonmuster von steigenden Flugpassagierzahlen in den Sommermonaten, gefolgt von Rückgängen in den Wintermonaten. Dies passt zu der Vorstellung, dass der Sommer eine beliebte Zeit für Urlaub und Reisen ist. Wir sehen auch, dass diese Saisonzyklen im Verlauf der Zeitreihe erheblich an Amplitude zunehmen.

Restkomponente

Das Diagramm für die Restkomponente zeigt alle Informationen, die nicht in der Trend- und Saisonzerlegung erfasst wurden. Die Restkomponente enthält statistisches Rauschen, kann aber auf eine falsche Einstellung der Argumente der STL-Trend- und -Saison-Funktion hinweisen. In der Regel sind periodische Schwankungen in der Restkomponente des Signals oder eindeutig nicht zufällig angezeigte Informationen ein Zeichen dafür, dass in der Zeitreihe Informationen vorhanden sind, die aktuell nicht in der Saison- oder der Trendkomponente erfasst werden. In diesem Fall müssen Sie Ihre Definitionen der einzelnen Funktionsargumente überprüfen und ggf. die Periodizität ändern.

Glättungswerte

Da wir keine Werte für die Trend- und Saisonglättung angegeben haben, verwendet die Funktion die Standardwerte für diese Parameter. In Qlik Sense ergeben die Standardglättungswerte im STL-Algorithmus effektive Ergebnisse. Infolgedessen können diese Argumente in den meisten Fällen in den Formeln weggelassen werden.

Wenn für die Saison- oder Trendglättungsargumente 0 in einer der drei STL-Funktionen festgelegt wird, verwendet der Algorithmus Standardwerte anstelle von 0-Werten.

Der Trendglättungswert verwendet die Dimension, die im Diagramm angegeben ist. Da das Feld *YearMonth* Daten nach Monaten angibt, ist der Trendglättungswert die Anzahl der Monate. Die Saisonglättung entspricht der definierten Periodizität. Da wir in diesem Fall einen Zeitraum mit einer Dauer von zwölf Monaten (einem Jahr) definiert haben, ist der Saisonglättungswert die Anzahl der Jahre. Das kann verwirrend klingen, bedeutet aber tatsächlich, dass wir zum Ermitteln der Saisonalität eine Anzahl von Saisonen betrachten müssen. Diese Anzahl ist die Saisonglättung.

Weitere nützliche Informationen

Da die Amplitude der Saisonzyklen im Lauf der Zeit zunimmt, kann ein erweiterter Analyseansatz logarithmische Funktionen nutzen, um eine multiplikative Zerlegung zu erstellen. In der Praxis kann eine einfache Kennzahl mit relativer Amplitude in Qlik Sense erstellt werden, indem die Saisonkomponente durch die Trendkomponente geteilt wird. Danach stellen wir fest, dass im Lauf der Zeit die Sommerspitzen jedes Zyklus eine größere relative Amplitude aufweisen. Die Amplitude der Tiefstände im Winter nimmt jedoch im Zeitverlauf nicht zu.

8.23 Verteilungsfunktionen

Statistische Verteilungsfunktionen geben die Wahrscheinlichkeit des Auftretens verschiedener möglicher Ergebnisse für eine bestimmte Eingabevariable zurück. Sie können diese Funktionen verwenden, um die potenziellen Werte Ihrer Datenpunkte zu berechnen.

Die nachfolgend beschriebenen drei Gruppen von Funktionen zur statistischen Verteilung sind in Qlik Sense implementiert und nutzen die Cephes-Funktionsbibliothek. Referenzen und Details über die verwendeten Algorithmen, Genauigkeiten usw. finden Sie unter: Cephes library. Die Cephes-Bibliothek wird mit Genehmigung verwendet.

- Die Wahrscheinlichkeitsfunktionen berechnen die Wahrscheinlichkeit an dem Punkt in der Verteilung, der von dem angegebenen Wert bezeichnet wird.
    - Die Häufigkeitsfunktionen werden für diskrete Verteilungen verwendet.
    - Die Dichtefunktionen werden für fortlaufende Funktionen verwendet.
- Die Dist-Funktionen berechnen die kumulierte Wahrscheinlichkeit der Verteilung an dem Punkt in der Verteilung, der von dem angegebenen Wert bezeichnet wird.
- Die Inv-Funktionen berechnen den umgekehrten Wert anhand der kumulierten Wahrscheinlichkeit der Verteilung.

Alle Funktionen können sowohl im Datenladeskript als auch in den Diagrammformeln verwendet werden.

Verteilungsfunktionen für statistische Tests – Übersicht

Jede Funktion wird nach der Übersicht genauer beschrieben. Sie können auch auf den Funktionsnamen in der Syntax klicken, um direkt auf die Details zu der spezifischen Funktion zuzugreifen.

BetaDensity

`BetaDensity()` gibt die Wahrscheinlichkeit der Beta-Verteilung zurück.

```
BetaDensity (value, alpha, beta)
```

BetaDist

`BetaDist()` gibt die kumulierte Wahrscheinlichkeit der Beta-Verteilung zurück.

```
BetaDist (value, alpha, beta)
```

BetaInv

`BetaINV()` gibt den Kehrwert der kumulierten Wahrscheinlichkeit der Beta-Verteilung zurück.

```
BetaInv (prob, alpha, beta)
```

BinomDist

`BinomDist()` gibt die kumulierte Wahrscheinlichkeit der Binomialverteilung zurück.

```
BinomDist (value, trials, trial_probability)
```

BinomFrequency

`BinomFrequency()` gibt die Binomial-Wahrscheinlichkeitsverteilung zurück.

```
BinomFrequency (value, trials, trial_probability)
```

BinomInv

`BinomInv()` gibt denKehrwert der kumulierten Wahrscheinlichkeit der Binomialverteilung zurück.

```
BinomInv (prob, trials, trial_probability)
```

ChiDensity

`ChiDensity()` gibt die einseitige Wahrscheinlichkeit der chi^2-Verteilung zurück. Die chi^2-Dichtefunktion ist einem chi^2-Test zugeordnet.

```
ChiDensity (value, degrees_freedom)
```

ChiDist

`ChiDist()` gibt die einseitige Wahrscheinlichkeit der chi^2-Verteilung zurück. Die chi^2-Verteilung ist einem chi^2-Test zugeordnet.

```
ChiDist (value, degrees_freedom)
```

ChiInv

`ChiInv()` liefert die Umkehrung der einseitigen chi^2-Verteilung.

```
ChiInv (prob, degrees_freedom)
```

FDensity

`FDensity()` gibt die Wahrscheinlichkeit der `F`-Verteilung zurück.

```
FDensity (value, degrees_freedom1, degrees_freedom2)
```

FDist

`FDist()` gibt die kumulierte Wahrscheinlichkeit der `F`-Verteilung zurück.

```
FDist (value, degrees_freedom1, degrees_freedom2)
```

FInv

`FInv()` gibt den Kehrwert der kumulierten Wahrscheinlichkeit der `F`-Verteilung zurück.

```
FInv (prob, degrees_freedom1, degrees_freedom2)
```

GammaDensity

`GammaDensity()` gibt die Wahrscheinlichkeit der Gamma-Verteilung zurück.

```
GammaDensity (value, k, θ)
```

GammaDist

`GammaDist()` gibt die kumulierte Wahrscheinlichkeit der Gamma-Verteilung zurück.

```
GammaDist (value, k, θ)
```

GammaInv

`GammaInv()` gibt den Kehrwert der kumulierten Wahrscheinlichkeit der Gamma-Verteilung zurück.

```
GammaInv (prob, k, θ)
```

NormDist

`NormDist()` liefert die kumulative Normalverteilung bei gegebenem Mittelwert und Standardabweichung. Ist mean = 0 und standard_dev = 1, liefert die Funktion die Standardnormalverteilung.

```
NormDist (value, mean, standard_dev)
```

NormInv

`NormInv()` liefert die Umkehrung der kumulativen Normalverteilung bei gegebenem Mittelwert und Standardabweichung.

```
NormInv (prob, mean, standard_dev)
```

PoissonDist

`PoissonDist()` gibt die kumulierte Wahrscheinlichkeit der Poisson-Verteilung zurück.

```
PoissonDist (value, mean)
```

PoissonFrequency

`PoissonFrequency()` gibt die Poisson-Wahrscheinlichkeitsverteilung zurück.

```
PoissonFrequency (value, mean)
```

PoissonInv

`PoissonInv()` gibt den Kehrwert der kumulierten Wahrscheinlichkeit der Poisson-Verteilung zurück.

```
PoissonInv (prob, mean)
```

TDensity

`TDensity()` gibt den Wert für die `t`-Test-Dichtefunktion zurück, wobei ein numerischer Wert ein berechneter Wert von `t` ist, dessen Wahrscheinlichkeit berechnet werden soll.

```
TDensity (value, degrees_freedom, tails)
```

TDist

`TDist()` gibt die Wahrscheinlichkeit der `t`-Verteilung zurück, wobei ein numerischer Wert ein berechneter Wert von `t` ist, dessen Wahrscheinlichkeit ermittelt werden soll.

```
TDist (value, degrees_freedom, tails)
```

TInv

`TInv()` gibt den `t`-Wert der `t`-Verteilung als Funktion der Wahrscheinlichkeit und Freiheitsgrade zurück.

```
TInv (prob, degrees_freedom)
```

Siehe auch:

Statistische Aggregierungsfunktionen (page 411)

BetaDensity

`BetaDensity()` gibt die Wahrscheinlichkeit der Beta-Verteilung zurück.

Syntax:

```
BetaDensity(value, alpha, beta)
```

Rückgabe Datentyp: Zahl

Argumente

Argument	Beschreibung
`value`	Der Wert, dessen Wahrscheinlichkeit Sie ermitteln wollen. Der Wert muss zwischen 0 und 1 liegen.
`alpha`	Eine positive Zahl, die den ersten Formparameter definiert. Sie ist der Exponent der Zufallsvariablen.
`beta`	Eine positive Zahl, die den zweiten Formparameter definiert. Sie gibt die Anzahl der Freiheitsgrade des Nenners an.

BetaDist

`BetaDist()` gibt die kumulierte Wahrscheinlichkeit der Beta-Verteilung zurück.

Syntax:

```
BetaDist(value, alpha, beta)
```

Rückgabe Datentyp: Zahl

Argumente

Argument	Beschreibung
`value`	Der Wert, dessen Wahrscheinlichkeit Sie ermitteln wollen. Der Wert muss zwischen 0 und 1 liegen.
`alpha`	Eine positive Zahl, die den ersten Formparameter definiert. Sie ist der Exponent der Zufallsvariablen.
`beta`	Eine positive Zahl, die den zweiten Formparameter definiert. Sie ist der Exponent, der die Form der Verteilung steuert.

Diese Funktion hängt auf folgende Weise mit der `BetaInv`-Funktion zusammen:
`If prob = BetaDist(value, alpha, beta), then BetaInv(prob, alpha, beta) = value`

BetaInv

`BetaINV()` gibt den Kehrwert der kumulierten Wahrscheinlichkeit der Beta-Verteilung zurück.

Syntax:

```
BetaInv(prob, alpha, beta)
```

Rückgabe Datentyp: Zahl

Argumente

Argument	Beschreibung
prob	Eine Wahrscheinlichkeit, die mit der Beta-Wahrscheinlichkeitsverteilung verknüpft ist. Es muss daher eine Zahl zwischen 0 und 1 sein.
alpha	Eine positive Zahl, die den ersten Formparameter definiert. Sie ist der Exponent der Zufallsvariablen.
beta	Eine positive Zahl, die den zweiten Formparameter definiert. Sie ist der Exponent, der die Form der Verteilung steuert.

Diese Funktion hängt auf folgende Weise mit der `BetaDist`-Funktion zusammen:
`If prob = BetaDist(value, alpha, beta), then BetaInv(prob, alpha, beta) = value`

BinomDist

`BinomDist()` gibt die kumulierte Wahrscheinlichkeit der Binomialverteilung zurück.

Syntax:

```
BinomDist(value, trials, trial_probability)
```

Rückgabe Datentyp: Zahl

Argumente

Argument	Beschreibung
value	Der Wert, dessen Wahrscheinlichkeit Sie ermitteln wollen. Der Wert muss eine Ganzzahl nicht kleiner als Null und nicht größer als die Anzahl der Versuche sein.
trials	Eine positive Ganzzahl, die die Anzahl der Versuche angibt.
trial_probability	Die Erfolgswahrscheinlichkeit für jeden Versuch. Sie ist immer eine Zahl zwischen 0 und 1.

Diese Funktion hängt auf folgende Weise mit der `BinomInv`-Funktion zusammen:
`If prob = BinomDIST(value, trials, trial_probability), then BinomInv(prob, trials, trial_probability) = value`

BinomFrequency

`BinomFrequency()` gibt die Binomial-Wahrscheinlichkeitsverteilung zurück.

Syntax:

```
BinomFrequency(value, trials, trial_probability)
```

Rückgabe Datentyp: Zahl

Argumente

Argument	Beschreibung
`value`	Der Wert, dessen Wahrscheinlichkeit Sie ermitteln wollen. Der Wert muss eine Ganzzahl nicht kleiner als Null und nicht größer als die Anzahl der Versuche sein.
`trials`	Eine positive Ganzzahl, die die Anzahl der Versuche angibt.
`trial_probability`	Die Erfolgswahrscheinlichkeit für jeden Versuch. Sie ist immer eine Zahl zwischen 0 und 1.

BinomInv

`BinomInv()` gibt denKehrwert der kumulierten Wahrscheinlichkeit der Binomialverteilung zurück.

Syntax:

```
BinomInv(prob, trials, trial_probability)
```

Rückgabe Datentyp: Zahl

Argumente

Argument	Beschreibung
`prob`	Eine Wahrscheinlichkeit, die mit der Binomial-Wahrscheinlichkeitsverteilung verknüpft ist. Es muss daher eine Zahl zwischen 0 und 1 sein.
`trials`	Eine positive Ganzzahl, die die Anzahl der Versuche angibt.
`trial_probability`	Die Erfolgswahrscheinlichkeit für jeden Versuch. Sie ist immer eine Zahl zwischen 0 und 1.

Diese Funktion hängt auf folgende Weise mit der `BinomDist`-Funktion zusammen:
`If prob = BinomDist(value, trials, trial_probability), then BinomInv(prob, trials, trial_probability) = value`

ChiDensity

`ChiDensity()` gibt die einseitige Wahrscheinlichkeit der chi^2-Verteilung zurück. Die chi^2-Dichtefunktion ist einem chi^2-Test zugeordnet.

Syntax:

```
ChiDensity(value, degrees_freedom)
```

Rückgabe Datentyp: Zahl

Argumente

Argument	Beschreibung
value	Der Wert, dessen Wahrscheinlichkeit Sie ermitteln wollen. Der Wert darf nicht negativ sein.
degrees_ freedom	Eine positive ganze Zahl, welche die Freiheitsgrade des Zählers angibt.

ChiDist

ChiDist() gibt die einseitige Wahrscheinlichkeit der chi^2-Verteilung zurück. Die chi^2-Verteilung ist einem chi^2-Test zugeordnet.

Syntax:

```
CHIDIST(value, degrees_freedom)
```

Rückgabe Datentyp: Zahl

Argumente:

Argumente

Argument	Beschreibung
value	Der Wert, dessen Wahrscheinlichkeit Sie ermitteln wollen. Der Wert darf nicht negativ sein.
degrees_ freedom	Eine positive ganze Zahl, welche die Freiheitsgrade angibt.

Diese Funktion hängt auf folgende Weise mit der **ChiInv**-Funktion zusammen:
`If prob = CHIDIST(value,df), then CHIINV(prob, df) = value`

Beschränkungen:

Alle Argumente müssen numerisch sein, sonst wird NULL ausgegeben.

Beispiele und Ergebnisse:

Beispiel	Ergebnis
CHIDIST(8, 15)	Liefert 0,9238

ChiInv

ChiInv() liefert die Umkehrung der einseitigen chi^2-Verteilung.

Syntax:

```
CHIINV(prob, degrees_freedom)
```

Rückgabe Datentyp: Zahl

Argumente:

Argumente

Argument	Beschreibung
prob	Die Wahrscheinlichkeit der chi^2-Verteilung. Es muss daher eine Zahl zwischen 0 und 1 sein.
degrees_ freedom	Eine ganze Zahl, welche die Freiheitsgrade angibt.

Diese Funktion hängt auf folgende Weise mit der **ChiDist**-Funktion zusammen:
`If prob = CHIDIST(value,df), then CHIINV(prob, df) = value`

Beschränkungen:

Alle Argumente müssen numerisch sein, sonst wird NULL ausgegeben.

Beispiele und Ergebnisse:

Beispiel	Ergebnis
CHIINV(0.9237827, 15)	Liefert 8,0000

FDensity

`FDensity()` gibt die Wahrscheinlichkeit der F-Verteilung zurück.

Syntax:

```
FDensity(value, degrees_freedom1, degrees_freedom2)
```

Rückgabe Datentyp: Zahl

Argumente

Argument	Beschreibung
`value`	Der Wert, dessen Wahrscheinlichkeit Sie ermitteln wollen. Der Wert darf nicht negativ sein.
`degrees_ freedom1`	Eine positive ganze Zahl, welche die Freiheitsgrade des Zählers angibt.
`degrees_ freedom2`	Eine positive ganze Zahl, welche die Freiheitsgrade des Nenners angibt.

FDist

`FDist()` gibt die kumulierte Wahrscheinlichkeit der F-Verteilung zurück.

Syntax:

```
FDist(value, degrees_freedom1, degrees_freedom2)
```

Rückgabe Datentyp: Zahl

Argumente:

Argumente

Argument	Beschreibung
value	Der Wert, dessen Wahrscheinlichkeit Sie ermitteln wollen. Der Wert darf nicht negativ sein.
degrees_ freedom1	Eine positive ganze Zahl, welche die Freiheitsgrade des Zählers angibt.
degrees_ freedom2	Eine positive ganze Zahl, welche die Freiheitsgrade des Nenners angibt.

Diese Funktion hängt auf folgende Weise mit der **FInv**-Funktion zusammen:
`If prob = FDIST(value, df1, df2), then FINV(prob, df1, df2) = value`

Beschränkungen:

Alle Argumente müssen numerisch sein, sonst wird NULL ausgegeben.

Beispiele und Ergebnisse:

Beispiel	Ergebnis
FDIST(15, 8, 6)	Liefert 0,0019

FInv

`FInv()` gibt den Kehrwert der kumulierten Wahrscheinlichkeit der F-Verteilung zurück.

Syntax:

```
FInv(prob, degrees_freedom1, degrees_freedom2)
```

Rückgabe Datentyp: Zahl

Argumente:

Argumente

Argument	Beschreibung
prob	Die Wahrscheinlichkeit der F-Wahrscheinlichkeitsverteilung, und demnach eine Zahl zwischen 0 und 1.
degrees_ freedom	Eine ganze Zahl, welche die Freiheitsgrade angibt.

Diese Funktion hängt auf folgende Weise mit der **FDist**-Funktion zusammen:
`If prob = FDIST(value, df1, df2), then FINV(prob, df1, df2) = value`

Beschränkungen:

Alle Argumente müssen numerisch sein, sonst wird NULL ausgegeben.

Beispiele und Ergebnisse:

Beispiel	Ergebnis
FINV(0.0019369, 8, 6)	Liefert 15,0000

GammaDensity

`GammaDensity()` gibt die Wahrscheinlichkeit der Gamma-Verteilung zurück.

Syntax:

```
GammaDensity(value, k, θ)
```

Rückgabe Datentyp: Zahl

Argumente

Argument	Beschreibung
value	Der Wert, dessen Wahrscheinlichkeit Sie ermitteln wollen. Der Wert darf nicht negativ sein.
k	Eine positive Zahl, die den Formparameter definiert.
θ	Eine positive Zahl, die den Skalenparameter definiert.

GammaDist

`GammaDist()` gibt die kumulierte Wahrscheinlichkeit der Gamma-Verteilung zurück.

Syntax:

```
GammaDist(value, k, θ)
```

Rückgabe Datentyp: Zahl

Argumente

Argument	Beschreibung
value	Der Wert, dessen Wahrscheinlichkeit Sie ermitteln wollen. Der Wert darf nicht negativ sein.
k	Eine positive Zahl, die den Formparameter definiert.
θ	Eine positive Zahl, die den Skalenparameter definiert.

Diese Funktion hängt auf folgende Weise mit der `GammaINV`-Funktion zusammen:
`If prob = GammaDist(value, k, θ), then GammaInv(prob, k, θ) = value`

GammaInv

`GammaInv()` gibt den Kehrwert der kumulierten Wahrscheinlichkeit der Gamma-Verteilung zurück.

Syntax:

```
GammaInv(prob, k, θ)
```

Rückgabe Datentyp: Zahl

Argumente

Argument	Beschreibung
prob	Eine Wahrscheinlichkeit, die mit der Gamma-Wahrscheinlichkeitsverteilung verknüpft ist. Es muss daher eine Zahl zwischen 0 und 1 sein.
k	Eine positive Zahl, die den Formparameter definiert.
θ	Eine positive Zahl, die den Skalenparameter definiert.

Diese Funktion hängt auf folgende Weise mit der `GammaDist`-Funktion zusammen:
`If prob = GammaDist(value, k, θ), then GammaInv(prob, k, θ) = value`

NormDist

`NormDist()` liefert die kumulative Normalverteilung bei gegebenem Mittelwert und Standardabweichung. Ist mean = 0 und standard_dev = 1, liefert die Funktion die Standardnormalverteilung.

Syntax:

```
NORMDIST(value, [mean], [standard_dev], [cumulative])
```

Rückgabe Datentyp: Zahl

Argumente:

Argumente

Argument	Beschreibung
value	Der Wert, dessen Wahrscheinlichkeit Sie ermitteln wollen.
mean	Optionaler Wert, der das arithmetische Mittel der Verteilung angibt. Wenn Sie dieses Argument nicht angeben, lautet der Standardwert 0.
standard_dev	Optionaler positiver Wert, der die Standardabweichung der Verteilung angibt. Wenn Sie dieses Argument nicht angeben, lautet der Standardwert 1.

Argument	Beschreibung
cumulative	Sie können optional eine Standardnormalverteilung oder eine kumulative Verteilung auswählen. 0 = Standardnormalverteilung 1 = kumulative Verteilung (Standard)

Diese Funktion hängt auf folgende Weise mit der **NormInv**-Funktion zusammen:
`If prob = NORMDIST(value, m, sd), then NORMINV(prob, m, sd) = value`

Beschränkungen:

Alle Argumente müssen numerisch sein, sonst wird NULL ausgegeben.

Beispiele und Ergebnisse:

Beispiel	Ergebnis
NORMDIST(0.5, 0, 1)	Liefert 0,6915

NormInv

`NormInv()` liefert die Umkehrung der kumulativen Normalverteilung bei gegebenem Mittelwert und Standardabweichung.

Syntax:

```
NORMINV(prob, mean, standard_dev)
```

Rückgabe Datentyp: Zahl

Argumente:

Argumente

Argument	Beschreibung
prob	Die Wahrscheinlichkeit der Normalverteilung. Es muss daher eine Zahl zwischen 0 und 1 sein.
mean	Das arithmetische Mittel der Verteilung.
standard_ dev	Eine positive Zahl, welche die Standardabweichung der Verteilung angibt.

Diese Funktion hängt auf folgende Weise mit der **NormDist**-Funktion zusammen:
`If prob = NORMDIST(value, m, sd), then NORMINV(prob, m, sd) = value`

Beschränkungen:

Alle Argumente müssen numerisch sein, sonst wird NULL ausgegeben.

Beispiele und Ergebnisse:

Beispiel	Ergebnis
NORMINV(0.6914625, 0, 1)	Liefert 0,5000

PoissonDist

`PoissonDist()` gibt die kumulierte Wahrscheinlichkeit der Poisson-Verteilung zurück.

Syntax:

```
PoissonDist(value, mean)
```

Rückgabe Datentyp: Zahl

Argumente

Argument	Beschreibung
`value`	Der Wert, dessen Wahrscheinlichkeit Sie ermitteln wollen. Der Wert darf nicht negativ sein.
`mean`	Eine positive Zahl, die das durchschnittliche Ergebnis definiert.

Diese Funktion hängt auf folgende Weise mit der `PoissonInv`-Funktion zusammen:
`If prob = PoissonDist(value, mean), then PoissonInv(prob, mean) = value`

PoissonFrequency

`PoissonFrequency()` gibt die Poisson-Wahrscheinlichkeitsverteilung zurück.

Syntax:

```
PoissonFrequency(value, mean)
```

Rückgabe Datentyp: Zahl

Argumente

Argument	Beschreibung
`value`	Der Wert, dessen Wahrscheinlichkeit Sie ermitteln wollen. Der Wert darf nicht negativ sein.
`mean`	Eine positive Zahl, die das durchschnittliche Ergebnis definiert.

PoissonInv

`PoissonInv()` gibt den Kehrwert der kumulierten Wahrscheinlichkeit der Poisson-Verteilung zurück.

Syntax:

```
PoissonInv(prob, mean)
```

Rückgabe Datentyp: Zahl

Argumente

Argument	Beschreibung
prob	Eine Wahrscheinlichkeit, die mit der Poisson-Wahrscheinlichkeitsverteilung verknüpft ist. Es muss daher eine Zahl zwischen 0 und 1 sein.
mean	Eine positive Zahl, die das durchschnittliche Ergebnis definiert.

Diese Funktion hängt auf folgende Weise mit der PoissonDIST-Funktion zusammen:
`If prob = PoissonDist(value, mean), then PoissonInv(prob, mean) = value`

TDensity

`TDensity()` gibt den Wert für die t-Test-Dichtefunktion zurück, wobei ein numerischer Wert ein berechneter Wert von `t` ist, dessen Wahrscheinlichkeit berechnet werden soll.

Syntax:

```
TDensity(value, degrees_freedom)
```

Rückgabe Datentyp: Zahl

Argumente

Argument	Beschreibung
value	Der Wert, dessen Wahrscheinlichkeit Sie ermitteln wollen. Der Wert darf nicht negativ sein.
degrees_ freedom	Eine positive ganze Zahl, welche die Freiheitsgrade angibt.

TDist

`TDist()` gibt die Wahrscheinlichkeit der t-Verteilung zurück, wobei ein numerischer Wert ein berechneter Wert von `t` ist, dessen Wahrscheinlichkeit ermittelt werden soll.

Syntax:

```
TDist(value, degrees_freedom, tails)
```

Rückgabe Datentyp: Zahl

Argumente:

Argumente

Argument	Beschreibung
value	Der Wert, dessen Wahrscheinlichkeit Sie ermitteln wollen. Der Wert darf nicht negativ sein.
degrees_ freedom	Eine positive ganze Zahl, welche die Freiheitsgrade angibt.
tails	Ist entweder 1 (einseitige Verteilung) oder 2 (zweiseitige Verteilung).

Diese Funktion hängt auf folgende Weise mit der **TInv**-Funktion zusammen:
`If prob = TDIST(value, df ,2), then TINV(prob, df) = value`

Beschränkungen:

Alle Argumente müssen numerisch sein, sonst wird NULL ausgegeben.

Beispiele und Ergebnisse:

Beispiel	Ergebnis
TDIST(1, 30, 2)	Liefert 0,3253

TInv

`TInv()` gibt den `t`-Wert der `t`-Verteilung als Funktion der Wahrscheinlichkeit und Freiheitsgrade zurück.

Syntax:

```
TINV(prob, degrees_freedom)
```

Rückgabe Datentyp: Zahl

Argumente:

Argumente

Argument	Beschreibung
prob	Die Wahrscheinlichkeit der zweiseitigen t-Verteilung. Es muss daher eine Zahl zwischen 0 und 1 sein.
degrees_ freedom	Eine ganze Zahl, welche die Freiheitsgrade angibt.

Beschränkungen:

Alle Argumente müssen numerisch sein, sonst wird NULL ausgegeben.

Diese Funktion hängt auf folgende Weise mit der **TDist**-Funktion zusammen:
`If prob = TDIST(value, df ,2), then TINV(prob, df) = value.`

Beispiele und Ergebnisse:

Beispiel	Ergebnis
TINV(0.3253086, 30)	Liefert 1,0000

8.24 Stringfunktionen

Dieser Abschnitt beschreibt Funktionen, mit denen Strings gehandhabt und beeinflusst werden.

Alle Funktionen können sowohl im Datenladeskript als auch in den Diagrammformeln verwendet werden, außer **Evaluate**, das nur im Datenladeskript verwendet werden kann.

Stringfunktionen – Übersicht

Jede Funktion wird nach der Übersicht genauer beschrieben. Sie können auch auf den Funktionsnamen in der Syntax klicken, um direkt auf die Details zu der spezifischen Funktion zuzugreifen.

Capitalize
Capitalize() liefert den String mit allen Wörtern mit großen Anfangsbuchstaben.

```
Capitalize (text)
```

Chr
Chr() liefert das dem eingegebenen Ganzzahlwert entsprechende Unicode-Zeichen.

```
Chr (int)
```

Evaluate
Evaluate() ermittelt, ob der eingegebene Textstring als gültige Qlik Sense-Formel evaluiert werden kann. Ist dies der Fall, wird der Wert der Formel als String zurückgegeben. Ist der eingegebene String keine gültige Formel, liefert diese Funktion NULL.

```
Evaluate (expression_text)
```

FindOneOf
FindOneOf() durchsucht einen String, um die Position des Vorkommens eines beliebigen Zeichens aus einem Satz von bereitgestellten Zeichen zu finden. Die Position des ersten Vorkommens eines beliebigen Zeichens aus dem Suchsatz wird ausgegeben, sofern kein drittes Argument (mit einem Wert über 1) angegeben wird. Wird keine Übereinstimmung gefunden, wird **0** zurückgegeben.

```
FindOneOf (text, char_set[, count])
```

Hash128

Hash128() liefert für jede Kombination von Formeln einen 128-Bit-Hash-Wert. Das Ergebnis ist ein 22-Zeichen-String.

```
Hash128 (expr{, expression})
```

Hash160

Hash160() liefert für jede Kombination von Formelwerten einen 160-Bit-Hash-Wert. Das Ergebnis ist ein 27-Zeichen-String.

```
Hash160 (expr{, expression})
```

Hash256

Hash256() liefert für jede Kombination von Formelwerten einen 256-Bit-Hash-Wert. Das Ergebnis ist ein 43-Zeichen-String.

```
Hash256 (expr{, expression})
```

Index

Index() durchsucht einen String, um die Startposition der n-ten Darstellung eines angegebenen Teilstrings zu ermitteln. Ein optionales drittes Argument liefert den Wert von n. Erfolgt keine Eingabe, wird 1 verwendet. Bei einem negativen Wert wird vom Ende des Strings aus gesucht. Die Nummerierung der Zeichen beginnt stets bei **1**.

```
Index  (text, substring[, count])
```

IsJson

IsJson() testet, ob ein angegebener String gültige JSON-Daten (JavaScript Object Notation) enthält. Sie können auch einen spezifischen JSON-Datentyp validieren.

```
IsJson (json [, type])
```

JsonGet

JsonGet() gibt den Pfad eines JSON-Datenstrings (JavaScript Object Notation) zurück. Die Daten müssen gültiges JSON sein, können aber weitere Leerzeichen oder neue Zeilen enthalten.

```
JsonGet (json, path)
```

JsonSet

JsonSet() ändert einen String, der JSON-Daten (JavaScript Object Notation) enthält. Es kann ein JSON-Wert mit dem neuen, vom Pfad angegebenen Speicherort festgelegt oder eingefügt werden. Die Daten müssen gültiges JSON sein, können aber weitere Leerzeichen oder neue Zeilen enthalten.

```
JsonSet(json, path, value)
```

KeepChar

KeepChar() liefert einen String bestehend aus dem ersten String, 'text', abzüglich der im zweiten String NICHT enthaltenen Zeichen, „keep_chars“.

```
KeepChar (text, keep_chars)
```

Left

Left() liefert einen String bestehend aus den ersten Zeichen des Eingabestrings (ganz links), bei dem die Zahl der Zeichen durch das zweite Argument bestimmt wird.

```
Left (text, count)
```

Len

Len() liefert die Länge des Eingabestrings.

```
Len (text)
```

LevenshteinDist

LevenshteinDist() gibt die Levenshtein-Entfernung zwischen zwei Zeichenfolgen zurück. Dies ist als Mindestzahl von Bearbeitungen mit einem Zeichen (Einfügungen, Löschungen oder Ersetzungen) definiert, die zum Ändern einer Zeichenfolge in eine andere erforderlich sind. Die Funktion ist für den Vergleich von Fuzzy-Zeichenfolgen nützlich.

```
LevenshteinDist (text1, text2)
```

Lower

Lower() konvertiert alle Zeichen im Eingabestring in Kleinbuchstaben.

```
Lower (text)
```

LTrim

LTrim() liefert den Eingabestring ohne führende Leerzeichen.

```
LTrim (text)
```

Mid

Mid() liefert den Teil des Eingabestrings, der bei der Position des durch das zweite Argument, 'start', definierten Zeichens beginnt, wobei die Stringlänge durch das dritte Argument 'count' definiert wird. Wird 'count' weggelassen, wird die Position des ersten Vorkommens ausgegeben. Die Nummerierung des ersten Zeichens im Eingabestring beginnt bei 1.

```
Mid (text, start[, count])
```

Ord

Ord() liefert die Unicode Codepointnummer des ersten Zeichens des Eingabestrings.

```
Ord (text)
```

PurgeChar

PurgeChar() liefert einen String, der alle Zeichen des Eingabestrings ('text') enthält, außer den Zeichen im String des zweiten Arguments ('remove_chars').

```
PurgeChar (text, remove_chars)
```

Repeat

Repeat() bildet einen String bestehend aus dem Eingabestring mit einer durch das zweite Argument definierten Anzahl von Wiederholungen.

```
Repeat (text[, repeat_count])
```

Replace

Replace() liefert den String s, wobei ein bestimmter Eingabestring bei jedem Vorkommen durch einen anderen ersetzt wird. Es erfolgt nur ein Durchlauf von links nach rechts.

```
Replace (text, from_str, to_str)
```

Right

Right() liefert einen String bestehend aus den letzten (ganz rechts stehenden) Zeichen des Eingabestrings, wobei die Zahl der Zeichen durch das zweite Argument bestimmt wird.

```
Right (text, count)
```

RTrim

RTrim() liefert den Eingabestring ohne abschließende Leerzeichen.

```
RTrim (text)
```

SubField

SubField() wird zur Extrahierung von Teilstring-Komponenten aus einem übergeordneten Stringfeld verwendet, bei dem die Ursprungsdatensatzfelder aus zwei oder mehr Teilen bestehen, die durch ein Trennzeichen getrennt sind.

```
SubField (text, delimiter[, field_no ])
```

SubStringCount

SubStringCount() liefert die Anzahl der Vorkommen des angegebenen Teilstrings im Text des Eingabestrings. Gibt es keine Übereinstimmung, ist das Ergebnis 0.

```
SubStringCount (text, substring)
```

TextBetween

TextBetween() liefert den Text im Eingabestring, der zwischen den Zeichen angezeigt wird, die als Trennzeichen festgelegt wurden.

```
TextBetween (text, delimiter1, delimiter2[, n])
```

Trim

Trim() liefert den Eingabestring ohne führende und abschließende Leerzeichen.

```
Trim (text)
```

Upper

Upper() wandelt alle Zeichen im Eingabestring für alle Textzeichen in der Formel in Großbuchstaben um. Zahlen und Symbole werden ignoriert.

```
Upper (text)
```

Capitalize

Capitalize() liefert den String mit allen Wörtern mit großen Anfangsbuchstaben.

Syntax:

```
Capitalize(text)
```

Rückgabe Datentyp: String

Beispiel: Diagrammformeln

Beispiel	Ergebnis
`Capitalize ( 'star trek' )`	Liefert 'Star Trek'
`Capitalize ( 'AA bb cC Dd')`	Liefert 'Aa Bb Cc Dd'

Beispiel: Ladeskript

```
Load  String, Capitalize(String) Inline [String rHode iSland washingTon d.C. new york];
```

Ergebnis

String	Capitalize(String)
rHode iSland	Rhode Island
washingTon d.C.	Washington D.C.
new york	New York

Chr

Chr() liefert das dem eingegebenen Ganzzahlwert entsprechende Unicode-Zeichen.

Syntax:

```
Chr(int)
```

Rückgabe Datentyp: String

Beispiele und Ergebnisse:

Beispiel	Ergebnis
Chr(65)	Liefert den String 'A'
Chr(163)	Liefert den String '£'
Chr(35)	Liefert den String '#'

Evaluate

Evaluate() ermittelt, ob der eingegebene Textstring als gültige Qlik Sense-Formel evaluiert werden kann. Ist dies der Fall, wird der Wert der Formel als String zurückgegeben. Ist der eingegebene String keine gültige Formel, liefert diese Funktion NULL.

Syntax:

```
Evaluate(expression_text)
```

Rückgabe Datentyp: dual

Diese Stringfunktion kann in Diagrammformeln nicht verwendet werden.

Beispiele und Ergebnisse:

Funktionsbeispiel	Ergebnis
`Evaluate ( 5 * 8 )`	Liefert '40'

Ladeskriptbeispiel

```
Load  Evaluate(String) as Evaluated, String Inline [String 4 5+3 0123456789012345678 Today()
];
```

Ergebnis

String	Bewertet
4	4
5+3	8
0123456789012345678	0123456789012345678
Today()	2022-02-02

FindOneOf

FindOneOf() durchsucht einen String, um die Position des Vorkommens eines beliebigen Zeichens aus einem Satz von bereitgestellten Zeichen zu finden. Die Position des ersten Vorkommens eines beliebigen Zeichens aus dem Suchsatz wird ausgegeben, sofern kein drittes Argument (mit einem Wert über 1) angegeben wird. Wird keine Übereinstimmung gefunden, wird **0** zurückgegeben.

Syntax:

```
FindOneOf(text, char_set[, count])
```

Rückgabe Datentyp: ganze Zahl

Argumente:

Argumente

Argument	Beschreibung
text	Der Original-String.
char_set	Mehrere Zeichen, nach denen in text gesucht werden soll.
count	Definiert, nach welchem Vorkommen eines der Zeichen gesucht werden soll. Beim Wert 2 wird beispielsweise nach dem zweiten Vorkommen gesucht.

Beispiel: Diagrammformeln

Beispiel	Ergebnis
`FindOneOf( 'my example text string', 'et%s')`	Liefert '4', da ‘e’ das vierte Zeichen in der Beispielzeichenfolge ist.
`FindOneOf( 'my example text string', 'et%s', 3)`	Liefert '12', da die Suche nach einem beliebigen der Zeichen e, t, % oder s erfolgt und "t" das dritte Vorkommen ist und sich an Position 12 der Beispielzeichenfolge befindet.
`FindOneOf( 'my example text string', '¤%&')`	Liefert '0', da keines der Zeichen ¤, % oder & in der Beispielzeichenfolge vorkommt.

Beispiel: Ladeskript

```
Load * Inline [SearchFor, Occurrence et%s,1 et%s,3 ¤%&,1]
```

Ergebnis

SearchFor	Occurrence	FindOneOf('mein beispieltextstring', SearchFor, Occurrence)
et%s	1	4
et%s	3	12
¤%&	1	0

Hash128

Hash128() liefert für jede Kombination von Formeln einen 128-Bit-Hash-Wert. Das Ergebnis ist ein 22-Zeichen-String.

Syntax:

```
Hash128(expr{, expression})
```

Rückgabe Datentyp: String

Beispiel: Diagrammformeln

Beispiel	Ergebnis
Hash128 ('abc', 'xyz', '123')	Gibt „MA&5]6+3=:>:>G%S<U*S2+“ zurück.
Hash128 (Region, Year, Month) Note: Region, Year, and Month are table fields.	Gibt „G7*=6GKPJ(Z+)^KM?<$'A+“ zurück.

Beispiel: Ladeskript

```
Hash_128: Load *,  Hash128(Region, Year, Month) as Hash128; Load * inline [ Region, Year, Month abc, xyz, 123 EU, 2022, 01 UK, 2022, 02 US, 2022, 02 ];
```

Ergebnis

Region	Jahr	Monat	Hash128
abc	xyz	123	MA&5]6+3=:>;>G%S<U*S2+
EU	2022	01	B40^K&[T@!;VB'XR]<5=/$
UK	2022	02	O5T;+1?[B&"F&1//MA[MN!
US	2022	02	C6@#]4#_G-(]J7EQY#KRW0

Hash160

Hash160() liefert für jede Kombination von Formelwerten einen 160-Bit-Hash-Wert. Das Ergebnis ist ein 27-Zeichen-String.

Syntax:

```
Hash160(expr{, expression})
```

Rückgabe Datentyp: String

Beispiel: Diagrammformeln

Beispiel	Ergebnis
Hash160 ('abc', 'xyz', '123')	Gibt „MA&5]6+3=:>;>G%S<U*S2I:`=X*“ zurück.
Hash160 (Region, Year, Month) Note: Region, Year, and Month are table fields.	Gibt „G7*=6GKPJ (Z+)^KM?<$'AI.)?U$“ zurück.

Beispiel: Ladeskript

```
Hash_160: Load *,  Hash160(Region, Year, Month) as Hash160; Load * inline [ Region, Year,
Month abc, xyz, 123 EU, 2022, 01 UK, 2022, 02 US, 2022, 02 ];
```

Ergebnis

Region	Jahr	Monat	Hash160
abc	xyz	123	MA&5]6+3=:>;>G%S<U*S2I:`=X*
EU	2022	01	B40^K&[T@!;VB'XR]<5=//_F853
UK	2022	02	O5T;+1?[B&"F&1//MA[MN!T"FWZ
US	2022	02	C6@#]4#_G-(]J7EQY#KRW`@KF+W

Hash256

Hash256() liefert für jede Kombination von Formelwerten einen 256-Bit-Hash-Wert. Das Ergebnis ist ein 43-Zeichen-String.

Syntax:

```
Hash256(expr{, expression})
```

Rückgabe Datentyp: String

Beispiel: Diagrammformeln

Beispiel	Ergebnis
Hash256 ('abc', 'xyz', '123')	Gibt „MA&5]6+3=:>;>G%S<U*S2I:`=X*A.IO*8N\%Y7Q;YEJ“ zurück.
Hash256 (Region, Year, Month) Note: Region, Year, and Month are table fields.	Gibt „G7*=6GKPJ(Z+)^KM?<$'AI.)?U$#X2RB [:0ZP=+Z`F:“ zurück.

Beispiel: Ladeskript

```
Hash_256: Load *,  Hash256(Region, Year, Month) as Hash256; Load * inline [ Region, Year,
Month abc, xyz, 123 EU, 2022, 01 UK, 2022, 02 US, 2022, 02 ];
```

Ergebnis

Region	Jahr	Monat	Hash256
abc	xyz	123	MA&5]6+3=:>;>G%S<U*S2I:`=X*A.IO*8N\%Y7Q;YEJ
EU	2022	01	B40^K&[T@!;VB'XR]<5=//_F853?BE6'G&,YH*T'MF)
UK	2022	02	O5T;+1?[B&"F&1//MA[MN!T"FWZT=4\#V`M%6_\0C>4
US	2022	02	C6@#]4#_G-(]J7EQY#KRW`@KF+W-0]`[Z8R+#'")=+0

Index

Index() durchsucht einen String, um die Startposition der n-ten Darstellung eines angegebenen Teilstrings zu ermitteln. Ein optionales drittes Argument liefert den Wert von n. Erfolgt keine Eingabe, wird 1 verwendet. Bei einem negativen Wert wird vom Ende des Strings aus gesucht. Die Nummerierung der Zeichen beginnt stets bei **1**.

Syntax:

```
Index(text, substring[, count])
```

Rückgabe Datentyp: ganze Zahl

Argumente:

Argumente

Argument	Beschreibung
text	Der Original-String.
substring	Mehrere Zeichen, nach denen in text gesucht werden soll.
count	Definiert, nach welchem Vorkommen von **substring** gesucht werden soll. Beim Wert 2 wird beispielsweise nach dem zweiten Vorkommen gesucht.

Beispiele und Ergebnisse:

Beispiel	Ergebnis
Index('abcdefg', 'cd')	Liefert 3
Index('abcdabcd', 'b', 2)	Liefert 6 (das zweite Vorkommen von 'b')
Index('abcdabcd', 'b',-2)	Liefert 2 (das zweite Vorkommen von 'b', vom Ende aus gerechnet)
`Left( Date, Index( Date,'-') -1 )` where **Date** = 1997-07-14	Liefert 1997
`Mid( Date, Index( Date, '-', 2 ) -2, 2 )` where **Date** = 1997-07-14	Liefert 07

Beispiel: Skript

```
T1: Load  *,  index(String, 'cd') as Index_CD,              // returns 3  in Index_CD index
(String, 'b') as Index_B,               // returns 2 in Index_B index(String, 'b', -1) as
Index_B2;          // returns 2 or 6 in Index_B2 Load * inline [ String abcdefg abcdabcd ];
```

IsJson

IsJson() testet, ob ein angegebener String gültige JSON-Daten (JavaScript Object Notation) enthält. Sie können auch einen spezifischen JSON-Datentyp validieren.

Syntax:

```
value IsJson(json [, type])
```

Rückgabe Datentyp: dual

Argumente

Argument	Beschreibung
json	Zu testender String. Er kann zusätzliche Leerzeichen oder Zeilenumbrüche enthalten.
type	Optionales Argument, das den zu testenden JSON-Datentyp angibt. • 'value' (Standard) • 'object' • 'array' • 'String' • 'number' • 'Boolean' • 'null'

Beispiel: Gültiges JSON und Typ

Beispiel	Ergebnis
`IsJson('null')`	Liefert -1 (true)
`IsJson('"abc"', 'value')`	Liefert -1 (true)
`IsJson('"abc"', 'string')`	Liefert -1 (true)
`IsJson(123, 'number')`	Liefert -1 (true)

Beispiel: Ungültiges JSON oder Typ

Beispiel	Ergebnis	Beschreibung
`IsJson('text')`	Liefert 0 (false)	'text' ist kein gültiger JSON-Wert
`IsJson('"text"', 'number')`	Liefert 0 (false)	'"text"' ist keine gültige JSON-Zahl
`IsJson('"text"', 'text')`	Liefert 0 (false)	'text' ist kein gültiger JSON-Typ

JsonGet

JsonGet() gibt den Pfad eines JSON-Datenstrings (JavaScript Object Notation) zurück. Die Daten müssen gültiges JSON sein, können aber weitere Leerzeichen oder neue Zeilen enthalten.

Syntax:

```
value JsonGet(json, path)
```

Rückgabe Datentyp: dual

Argumente

Argument	Beschreibung
json	String mit JSON-Daten.
path	Der Pfad muss gemäß RFC 6901 angegeben werden. Dies ermöglicht ein Lookup von Eigenschaften innerhalb von JSON-Daten, ohne komplexe Substrings oder Indexfunktionen zu verwenden.

Beispiel: Gültiges JSON und Pfad

Beispiel	Ergebnis
`JsonGet('{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}', '')`	Liefert '{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}'
`JsonGet('{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}', '/a')`	Liefert '{"foo":"bar"}'
`JsonGet('{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}', '/a/foo')`	Liefert '"bar"'
`JsonGet('{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}', '/b')`	Liefert '[123,"abc","ABC"]'
`JsonGet('{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}', '/b/0')`	Liefert '123'
`JsonGet('{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}', '/b/1')`	Liefert '"abc"'
`JsonGet('{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}', '/b/2')`	Liefert '"ABC"'

Beispiel: Ungültiges JSON oder Pfad

Beispiel	Ergebnis	Beschreibung
`JsonGet('{"a":"b"}','/b')`	Liefert null	Der Pfad braucht nicht auf einen gültigen Teil der JSON-Daten zu zeigen.
`JsonGet('{"a"}','/a')`	Liefert null	Die JSON-Daten sind kein gültiges JSON (Mitglied „a“ hat keinen Wert).

JsonSet

JsonSet() ändert einen String, der JSON-Daten (JavaScript Object Notation) enthält. Es kann ein JSON-Wert mit dem neuen, vom Pfad angegebenen Speicherort festgelegt oder eingefügt werden. Die Daten müssen gültiges JSON sein, können aber weitere Leerzeichen oder neue Zeilen enthalten.

Syntax:

```
value JsonSet(json, path, value)
```

Rückgabe Datentyp: dual

Argumente

Argument	Beschreibung
json	String mit JSON-Daten.
path	Der Pfad muss gemäß RFC 6901 angegeben werden. Dies ermöglicht ein Buildup von Eigenschaften innerhalb von JSON-Daten, ohne komplexe Substrings oder Indexfunktionen und Verkettung zu verwenden.
value	Der neue String-Wert im JSON-Format.

Beispiel: Gültiges JSON, Pfad und Wert

Beispiel	Ergebnis
`JsonSet('{}','/a','"b"')`	Liefert '{"a":"b"}'
`JsonSet('[]','/0','"x"')`	Liefert '["x"]'
`JsonSet('"abc"','','123')`	Liefert 123

Beispiel: Gültiges JSON, Pfad oder Wert

Beispiel	Ergebnis	Beschreibung
`JsonSet ('"abc"','/x','123')`	Liefert null	Der Pfad braucht nicht auf einen gültigen Teil der JSON-Daten zu zeigen.
`JsonSet('{"a": {"b":"c"}}','a/b','"x"')`	Liefert null	Der Pfad ist ungültig.
`JsonSet(' {"a":"b"}','/a','abc')`	Liefert null	Der Wert ist kein gültiges JSON. Ein String muss in Anführungszeichen eingeschlossen sein.

KeepChar

KeepChar() liefert einen String bestehend aus dem ersten String, 'text', abzüglich der im zweiten String NICHT enthaltenen Zeichen, „keep_chars“.

Syntax:

```
KeepChar(text, keep_chars)
```

Rückgabe Datentyp: String

Argumente:

Argumente

Argument	Beschreibung
text	Der Original-String.
keep_chars	Ein String, der die Zeichen in text enthält, die behalten werden sollen.

Beispiel: Diagrammformeln

Beispiel	Ergebnis
`KeepChar ( 'a1b2c3','123' )`	Liefert '123'.
`KeepChar ( 'a1b2c3','1234' )`	Liefert '123'.
`KeepChar ( 'a1b22c3','1234' )`	Liefert '1223'.
`KeepChar ( 'a1b2c3','312' )`	Liefert '123'.

Beispiel: Ladeskript

```
T1:
Load
*,
keepchar(String1, String2) as KeepChar;
Load * inline [
String1, String2
'a1b2c3', '123'
];
```

Ergebnisse

Qlik Sense Tabelle mit der Ausgabe der *KeepChar*-Funktion im Ladeskript

String1	String2	KeepChar
a1b2c3	123	123

Siehe auch:

PurgeChar (page 1496)

Left

Left() liefert einen String bestehend aus den ersten Zeichen des Eingabestrings (ganz links), bei dem die Zahl der Zeichen durch das zweite Argument bestimmt wird.

Syntax:

```
Left(text, count)
```

Rückgabe Datentyp: String

Argumente:

Argument	Beschreibung
text	Der Original-String.
count	Legt die Zahl der Zeichen fest, die vom linken Teil des Strings **text** enthalten sein sollen.

Beispiel: Diagrammformel

Beispiel	Ergebnis
`Left('abcdef', 3)`	Liefert 'abc'

Beispiel: Ladeskript

```
T1: Load  *,  left(Text,Start) as Left;        Load * inline [ Text, Start 'abcdef', 3 '2021-07-14', 4 '2021-07-14', 2 ];
```

Ergebnis

Qlik Sense table showing the output from using the *Left* function in the load script.

Text	Start	Left
abcdef	3	abc
2021-07-14	4	2021
2021-07-14	2	20

Weitere Informationen finden Sie unter *Index (page 1486)*, was eine komplexere Stringanalyse ermöglicht.

Len

Len() liefert die Länge des Eingabestrings.

Syntax:

```
Len(text)
```

Rückgabe Datentyp: ganze Zahl

Beispiel: Diagrammformel

Beispiel	Ergebnis
`Len('Peter')`	Liefert '5'

Beispiel: Ladeskript

```
T1: Load String, First&Second as NewString; Load *, mid(String,len(First)+1) as Second; Load
*, upper(left(String,1)) as First; Load  * inline [ String this is a sample text string
capitalize first letter only ];
```

Ergebnis

String	NewString
this is a sample text string	This is a sample text string
capitalize first letter only	Capitalize first letter only

LevenshteinDist

LevenshteinDist() gibt die Levenshtein-Entfernung zwischen zwei Zeichenfolgen zurück. Dies ist als Mindestzahl von Bearbeitungen mit einem Zeichen (Einfügungen, Löschungen oder Ersetzungen) definiert, die zum Ändern einer Zeichenfolge in eine andere erforderlich sind. Die Funktion ist für den Vergleich von Fuzzy-Zeichenfolgen nützlich.

Syntax:

```
LevenshteinDist(text1, text2)
```

Rückgabe Datentyp: ganze Zahl

Beispiel: Diagrammformel

Beispiel	Ergebnis
`LevenshteinDist('Kitten','Sitting')`	Liefert 3

Beispiel: Ladeskript

Ladeskript

```
T1: Load *, recno() as ID; Load 'Silver' as String_1,* inline [ String_2 Sliver SSiver SSiveer
];  T1: Load *, recno()+3 as ID; Load 'Gold' as String_1,* inline [ String_2 Bold Bool Bond ];
 T1: Load *, recno()+6 as ID; Load 'Ove' as String_1,* inline [ String_2 Ove Uve Üve ];  T1:
Load *, recno()+9 as ID; Load 'ABC' as String_1,* inline [ String_2 DEFG abc ビビビ ];  set
nullinterpret = '<NULL>'; T1: Load *, recno()+12 as ID; Load 'X' as String_1,* inline [
String_2 '' <NULL> 1 ];  R1: Load  ID,  String_1,  String_2,  LevenshteinDist(String_1,
String_2) as LevenshteinDistance  resident T1;  Drop table T1;
```

Ergebnis

ID	String_1	String_2	LevenshteinDistance
1	Silver	Sliver	2
2	Silver	SSiver	2
3	Silver	SSiveer	3
4	Gold	Fett	1
5	Gold	Bool	3
6	Gold	Bond	2
7	Ove	Ove	0
8	Ove	Uve	1
9	Ove	Üve	1
10	ABC	DEFG	4
11	ABC	abc	3
12	ABC	ビビビ	3
13	X		1
14	X	-	1
15	X	1	1

Lower

Lower() konvertiert alle Zeichen im Eingabestring in Kleinbuchstaben.

Syntax:

```
Lower(text)
```

Rückgabe Datentyp: String

Beispiel: Diagrammformel

Beispiel	Ergebnis
`Lower('abcD')`	Liefert 'abcd'

Beispiel: Ladeskript

```
Load  String, Lower(String) Inline [String rHode iSland washingTon d.C. new york];
```

Ergebnis

String	Lower(String)
rHode iSland	rhode island
washingTon d.C.	washington d.c.
new york	new york

LTrim

LTrim() liefert den Eingabestring ohne führende Leerzeichen.

Syntax:

```
LTrim(text)
```

Rückgabe Datentyp: String

Beispiel: Diagrammformeln

Beispiel	Ergebnis
`LTrim( ' abc' )`	Liefert 'abc'
`LTrim( 'abc ' )`	Liefert 'abc '

Beispiel: Ladeskript

```
Set verbatim=1;  T1:  Load *, len(LtrimString) as LtrimStringLength;       Load *, ltrim
(String) as LtrimString; Load *, len(String) as StringLength;                Load * Inline [
String '   abc    ' ' def  '];
```

Die Anweisung „Set verbatim=1“ wird in das Beispiel eingeschlossen, um sicherzustellen, dass Leerzeichen nicht automatisch entfernt werden, bevor die ltrim-Funktion gezeigt wird. Weitere Informationen finden Sie unter Verbatim (page 214).

Ergebnis

String	StringLength	LtrimStringLength
def	6	5
abc	10	7

Siehe auch:

- *RTrim (page 1500)*

Mid

Mid() liefert den Teil des Eingabestrings, der bei der Position des durch das zweite Argument, 'start', definierten Zeichens beginnt, wobei die Stringlänge durch das dritte Argument 'count' definiert wird. Wird 'count' weggelassen, wird die Position des ersten Vorkommens ausgegeben. Die Nummerierung des ersten Zeichens im Eingabestring beginnt bei 1.

Syntax:

```
Mid(text, start[, count])
```

Rückgabe Datentyp: String

Argumente:

Argumente

Argument	Beschreibung
text	Der Original-String.
start	Ganzzahl, die die Position des ersten Zeichens in text festlegt, das enthalten sein soll.
count	Definiert die Stringlänge des Ausgabestrings. Ist nichts definiert, werden alle Zeichen ab der durch **start** definierten Position einbezogen.

Beispiel: Diagrammformeln

Beispiel	Ergebnis
`Mid('abcdef',3 )`	Liefert 'cdef'
`Mid('abcdef',3, 2 )`	Liefert 'cd'

Beispiel: Ladeskript

```
T1: Load *,  mid(Text,Start) as Mid1,            mid(Text,Start,Count) as Mid2;     Load *
inline [ Text, Start, Count 'abcdef', 3, 2 'abcdef', 2, 3 '210714', 3, 2 '210714', 2, 3 ];
```

Ergebnis

Qlik Sense table showing the output from using the *Mid* function in the load script.

Text	Start	Mid1	Anzahl	Mid2
abcdef	2	bcdef	3	bcd
abcdef	3	cdef	2	cd
210714	2	10714	3	107
210714	3	0714	2	07

Siehe auch:

Index (page 1486)

Ord

Ord() liefert die Unicode Codepointnummer des ersten Zeichens des Eingabestrings.

Syntax:

```
Ord(text)
```

Rückgabe Datentyp: ganze Zahl

Beispiele und Ergebnisse:

Beispiel: Diagrammformel

Beispiel	Ergebnis
`Ord('A')`	Liefert die Ganzzahl 65.
`Ord('Ab')`	Liefert die Ganzzahl 65.

Beispiel: Ladeskript

```
//Guqin (Chinese: 古琴) – 7-stringed zithers T2: Load *,  ord(Chinese) as OrdUnicode,
        ord(Western) as OrdASCII;                    Load * inline [ Chinese, Western 古琴,
Guqin ];
```

Ergebnis:

Chinesisch	Westlich	OrdASCII	OrdUnicode
古琴	Guqin	71	21476

PurgeChar

PurgeChar() liefert einen String, der alle Zeichen des Eingabestrings ('text') enthält, außer den Zeichen im String des zweiten Arguments ('remove_chars').

Syntax:

```
PurgeChar(text, remove_chars)
```

Rückgabe Datentyp: String

Argumente:

Argumente

Argument	Beschreibung
text	Der Original-String.
remove_chars	Ein String, der die Zeichen in text enthält, die entfernt werden sollen.

Rückgabe Datentyp: String

Beispiel: Diagrammformeln

Beispiel	Ergebnis
`PurgeChar ( 'a1b2c3','123' )`	Liefert „abc“.
`PurgeChar ( 'a1b2c3','312' )`	Liefert „abc“.

Beispiel: Ladeskript

```
T1:
Load
*,
purgechar(String1, String2) as PurgeChar;
Load * inline [
String1, String2
'a1b2c3', '123'
];
```

Ergebnisse

Qlik Sense Tabelle mit der Ausgabe der *PurgeChar*-Funktion im Ladeskript

String1	String2	PurgeChar
a1b2c3	123	abc

Siehe auch:

KeepChar (page 1489)

Repeat

Repeat() bildet einen String bestehend aus dem Eingabestring mit einer durch das zweite Argument definierten Anzahl von Wiederholungen.

Syntax:

```
Repeat(text[, repeat_count])
```

Rückgabe Datentyp: String

Argumente:

Argumente

Argument	Beschreibung
text	Der Original-String.
repeat_count	Legt fest, wie oft die Zeichen im String **text** im Ausgabestring wiederholt werden sollen.

Beispiel: Diagrammformel

Beispiel	Ergebnis
`Repeat( ' * ', rating )` when **rating** = 4	Liefert '****'

Beispiel: Ladeskript

```
T1: Load *, repeat(String,2) as Repeat; Load * inline [ String hello world! hOw aRe you? ];
```

Ergebnis

String	Wiederholen
hello world!	hello world!hello world!
hOw aRe you?	hOw aRe you?hOw aRe you?

Replace

Replace() liefert den String s, wobei ein bestimmter Eingabestring bei jedem Vorkommen durch einen anderen ersetzt wird. Es erfolgt nur ein Durchlauf von links nach rechts.

Syntax:

```
Replace(text, from_str, to_str)
```

Rückgabe Datentyp: String

Argumente:

Argumente

Argument	Beschreibung
text	Der Original-String.
from_str	Ein String, der einmal oder mehrmals innerhalb des Eingabestrings **text** vorkommen kann.
to_str	Der String, der alle Vorkommen von **from_str** im String **text** ersetzt.

Beispiele und Ergebnisse:

Beispiel	Ergebnis
`Replace('abccde','cc','xyz')`	Liefert 'abxyzde'

Siehe auch:

Right

Right() liefert einen String bestehend aus den letzten (ganz rechts stehenden) Zeichen des Eingabestrings, wobei die Zahl der Zeichen durch das zweite Argument bestimmt wird.

Syntax:

```
Right(text, count)
```

Rückgabe Datentyp: String

Argumente:

Argumente

Argument	Beschreibung
text	Der Original-String.
count	Legt ganz rechts beginnend die Zahl der Zeichen fest, die im String **text** enthalten sein sollen.

Beispiel: Diagrammformel

Beispiel	Ergebnis
`Right('abcdef', 3)`	Liefert 'def'

Beispiel: Ladeskript

```
T1:
Load
*,
right(Text,Start) as Right;
Load * inline [
Text, Start
'abcdef', 3
'2021-07-14', 4
'2021-07-14', 2
];
```

Ergebnis

Qlik Sense Tabelle mit der Ausgabe der *Right*-Funktion im Ladeskript

Text	Start	Right
abcdef	3	def
2021-07-14	4	7-14
2021-07-14	2	14

RTrim

RTrim() liefert den Eingabestring ohne abschließende Leerzeichen.

Syntax:

```
RTrim(text)
```

Rückgabe Datentyp: String

Beispiel: Diagrammformeln

Beispiel	Ergebnis
`RTrim( ' abc' )`	Liefert ' abc'
`RTrim( 'abc ' )`	Liefert 'abc'

Beispiel: Ladeskript

```
Set verbatim=1;    T1:  Load *, len(RtrimString) as RtrimStringLength;         Load *, rtrim
(String) as RtrimString; Load *, len(String) as StringLength;              Load * Inline [
String '   abc    ' ' def  '];
```

Die Anweisung „Set verbatim=1“ wird in das Beispiel eingeschlossen, um sicherzustellen, dass Leerzeichen nicht automatisch entfernt werden, bevor die rtrim-Funktion gezeigt wird. Weitere Informationen finden Sie unter Verbatim (page 214).

Ergebnis

String	StringLength	RtrimStringLength
def	6	4
abc	10	6

Siehe auch:

- *LTrim (page 1494)*

SubField

SubField() wird zur Extrahierung von Teilstring-Komponenten aus einem übergeordneten Stringfeld verwendet, bei dem die Ursprungsdatensatzfelder aus zwei oder mehr Teilen bestehen, die durch ein Trennzeichen getrennt sind.

Die Funktion **Subfield()** kann zum Beispiel verwendet werden, um Vor- und Nachnamen aus einer Liste von Datensätzen, die aus Vor- und Nachname bestehen, die Komponenten eines Pfadnamens oder Daten aus kommagetrennten Tabellen zu extrahieren.

Wenn Sie die Funktion **Subfield()** in einem **LOAD**-Befehl ohne den optionalen Parameter field_no verwenden, wird für jeden Teilstring ein vollständiger Datensatz generiert. Wenn mehrere Felder mit **Subfield()** geladen werden, werden die kartesischen Produkte aller Kombinationen erstellt.

Syntax:

```
SubField(text, delimiter[, field_no ])
```

Rückgabe Datentyp: String

Argumente:

Argumente

Argument	Beschreibung
text	Der Original-String. Dabei kann es sich um hartcodierten Text, eine Variable, eine Dollarzeichenerweiterung oder eine andere Formel handeln.
delimiter	Ein Zeichen in der Eingabe **text**, das den String in Komponenten aufteilt.
field_no	Das optionale dritte Argument ist eine Ganzzahl, die angibt, welcher der Teilstrings des übergeordneten Strings **text** geliefert werden soll. Verwenden Sie den Wert 1, um den ersten Teilstring zurückzugeben, den Wert 2, um den zweiten Teilstring zurückzugeben, usw. • Wenn **field_no** ein positiver Wert ist, werden Teilstrings von links nach rechts extrahiert. • Wenn **field_no** ein negativer Wert ist, werden Teilstrings von rechts nach links extrahiert.

SubField() kann anstelle komplexer Kombinationen von Funktionen wie Len(), Right(), Left(), Mid() und anderen Stringfunktionen verwendet werden.

Beispiele: Skript- und Diagrammformeln mit SubField

Beispiele – Skript- und Diagrammformeln

Einfache Beispiele

Beispiel	Ergebnis
`SubField(S, ';' ,2)`	Liefert 'cde', wenn **S** 'abc;cde;efg' ist.
`SubField(S, ';' ,1)`	Liefert einen leeren String, wenn **S** ein leerer String ist.
`SubField(S, ';' ,1)`	Liefert einen leeren String, wenn **S** ';' ist.
Nehmen wir an, Sie haben eine Variable, die einen Pfadnamen enthält, vMyPath, `Set vMyPath=\Users\ext_jrb\Documents\Qlik\Sense\Apps;.`	In einem Text- und Bilddiagramm können Sie eine Kennzahl wie `SubField(vMyPath, '\',-3)` hinzufügen, woraus sich „Qlik“ ergibt, weil dies der Substring an dritter Stelle vom rechten Ende der Variablen vMyPath ist.

Skriptbeispiel 1

Ladeskript

Laden Sie die folgenden Skriptformeln und Daten in den Dateneditor.

```
FullName:

LOAD * inline [
Name
'Dave Owen'
'Joe Tem'
];

SepNames:

Load Name,
SubField(Name, ' ',1) as FirstName,
SubField(Name, ' ',-1) as Surname
Resident FullName;
Drop Table FullName;
```

Erstellen einer Visualisierung

Erstellen Sie eine Tabellenvisualisierung in einem Qlik Sense Arbeitsblatt mit **Name**, **FirstName** und **SurName** als Dimensionen.

Ergebnis

Name	FirstName	SurName
Dave Owen	Dave	Owen
Joe Tem	Joe	Tem

Erläuterung

Die Funktion **SubField()** extrahiert den ersten Substring von **Name**, indem das Argument **field_no** auf 1 gesetzt wird. Da der Wert von **field_no** positiv ist, wird eine Reihenfolge von links nach rechts zum Extrahieren des Substrings verwendet. Ein zweiter Funktionsaufruf extrahiert den zweiten Substring, indem das Argument **field_no** auf -1 gesetzt wird. Dadurch wird der Substring in einer Reihenfolge von rechts nach links extrahiert.

Skriptbeispiel 2

Ladeskript

Laden Sie die folgenden Skriptformeln und Daten in den Dateneditor.

```
LOAD DISTINCT
Instrument,
SubField(Player,',') as Player,
SubField(Project,',') as Project;

Load * inline [
Instrument|Player|Project
Guitar|Neil,Mike|Music,Video
Guitar|Neil|Music,OST
Synth|Neil,Jen|Music,Video,OST
Synth|Jo|Music
Guitar|Neil,Mike|Music,OST
] (delimiter is '|');
```

Erstellen einer Visualisierung

Erstellen Sie eine Tabellenvisualisierung in einem Qlik Sense-Arbeitsblatt mit **Instrument**, **Player** und **Project** als Dimensionen.

Ergebnis

Instrument	Player	Project
Guitar	Mike	Music
Guitar	Mike	Video
Guitar	Mike	OST
Guitar	Neil	Music
Guitar	Neil	Video
Guitar	Neil	OST

Instrument	Player	Project
Synth	Jen	Music
Synth	Jen	Video
Synth	Jen	OST
Synth	Jo	Music
Synth	Neil	Music
Synth	Neil	Video
Synth	Neil	OST

Erläuterung

Dieses Beispiel zeigt, wie durch die Verwendung mehrerer Instanzen der Funktion **Subfield()** jeweils ohne den Parameter field_no aus demselben **LOAD**-Befehl heraus kartesische Produkte aller Kombinationen erstellt werden. Die Option **DISTINCT** dient dazu, zu verhindern, dass duplizierte Datensätze erstellt werden.

SubStringCount

SubStringCount() liefert die Anzahl der Vorkommen des angegebenen Teilstrings im Text des Eingabestrings. Gibt es keine Übereinstimmung, ist das Ergebnis 0.

Syntax:

```
SubStringCount(text, sub_string)
```

Rückgabe Datentyp: ganze Zahl

Argumente:

Argument	Beschreibung
text	Der Original-String.
sub_string	Ein String, der einmal oder mehrmals innerhalb des Eingabestrings **text** vorkommen kann.

Beispiel: Diagrammformeln

Beispiel	Ergebnis
`SubStringCount ( 'abcdefgcdxyz', 'cd' )`	Liefert '2'
`SubStringCount ( 'abcdefgcdxyz', 'dc' )`	Liefert '0'

Beispiel: Ladeskript

```
T1: Load *,  substringcount(upper(Strings),'AB') as SubStringCount_AB; Load * inline [ Strings
ABC:DEF:GHI:AB:CD:EF:GH aB/cd/ef/gh/Abc/abandoned ];
```

Ergebnis

Zeichenfolgen	SubStringCount_AB
aB/cd/ef/gh/Abc/abandoned	3
ABC:DEF:GHI:AB:CD:EF:GH	2

TextBetween

TextBetween() liefert den Text im Eingabestring, der zwischen den Zeichen angezeigt wird, die als Trennzeichen festgelegt wurden.

Syntax:

```
TextBetween(text, delimiter1, delimiter2[, n])
```

Rückgabe Datentyp: String

Argumente:

Argument	Beschreibung
text	Der Original-String.
delimiter1	Gibt das erste Trennzeichen (bzw. String) an, nach dem in **text** gesucht werden soll.
delimiter2	Gibt das zweite Trennzeichen (bzw. String) an, nach dem in **text** gesucht werden soll.
n	Definiert, zwischen welchem Vorkommen des Trennzeichenpaars gesucht werden soll. Ein Wert von 2 liefert beispielsweise die Zeichen zwischen dem zweiten Vorkommen des Trennzeichen1 und dem zweiten Vorkommen des Trennzeichen2.

Beispiel: Diagrammformeln

Beispiel	Ergebnis
`TextBetween('<abc>', '<', '>')`	Liefert 'abc'
`TextBetween('<abc><de>', '<', '>',2)`	Liefert 'de'
`TextBetween('abc', '<', '>')` `TextBetween('<a<b', '<', '>')`	Beide Beispiele liefern NULL. Wenn eines der Trennzeichen nicht in der Zeichenfolge gefunden wird, wird NULL geliefert.
`TextBetween('<>', '<', '>')`	Liefert eine Zeichenfolge mit Nulllänge.
`TextBetween('<abc>', '<', '>', 2)`	Liefert NULL, da n größer als die Anzahl der Vorkommen der Trennzeichen ist.

Beispiel: Ladeskript

```
Load *, textbetween(Text,'<','>') as TextBetween, textbetween(Text,'<','>',2) as
SecondTextBetween; Load * inline [ Text <abc><de> <def><ghi><jkl> ];
```

Ergebnis

Text	TextBetween	SecondTextBetween
<abc><de>	abc	de
<def><ghi><jkl>	def	ghi

Trim

Trim() liefert den Eingabestring ohne führende und abschließende Leerzeichen.

Syntax:

```
Trim(text)
```

Rückgabe Datentyp: String

Beispiele und Ergebnisse:

Beispiel: Diagrammformel

Beispiel	Ergebnis
`Trim( ' abc' )`	Liefert 'abc'
`Trim( 'abc ' )`	Liefert 'abc'
`Trim( ' abc ' )`	Liefert 'abc'

Beispiel: Ladeskript

```
Set verbatim=1;                                          T1: Load *, len(TrimString) as TrimStringLength;
(String) as TrimString; Load *, len(String) as StringLength;              Load * inline [
String '   abc    ' ' def  '](delimiter is '\t');
```

Die Anweisung „Set verbatim=1“ wird in das Beispiel eingeschlossen, um sicherzustellen, dass Leerzeichen nicht automatisch entfernt werden, bevor die trim-Funktion gezeigt wird. Weitere Informationen finden Sie unter Verbatim (page 214).

Ergebnis:

String	StringLength	TrimStringLength
def	6	3
abc	10	3

Upper

Upper() wandelt alle Zeichen im Eingabestring für alle Textzeichen in der Formel in Großbuchstaben um. Zahlen und Symbole werden ignoriert.

Syntax:

```
Upper(text)
```

Rückgabe Datentyp: String

Beispiel: Diagrammformel

Beispiel	Ergebnis
`Upper(' abcD')`	Liefert 'ABCD'

Beispiel: Ladeskript

```
Load  String,Upper(String) Inline [String rHode iSland washingTon d.C. new york];
```

Ergebnis

String	Upper(String)
rHode iSland	RHODE ISLAND
washingTon d.C.	WASHINGTON D.C.
new york	NEW YORK

8.25 Systemfunktionen

Systemfunktionen bieten Zugriff auf System-, Geräte- und Qlik Sense-App-Eigenschaften.

Systemfunktionen – Übersicht

Einige Funktionen werden nach der Übersicht genauer beschrieben. Bei diesen Funktionen können Sie auf den Funktionsnamen in der Syntax klicken, um direkt auf die Details zu der spezifischen Funktion zuzugreifen.

Author()

Diese Funktion liefert die Author-Eigenschaft der aktuellen App. Die Verwendung ist sowohl im Datenladeskript als auch in der Diagrammformel möglich.

Die Author-Eigenschaft kann in der aktuellen Version von Qlik Sense nicht festgelegt werden. Beim Migrieren eines QlikView-Dokuments wird die Author-Eigenschaft beibehalten.

ClientPlatform()

Diese Funktion liefert den Benutzer-Agenten des Client-Browsers. Die Verwendung ist sowohl im Datenladeskript als auch in der Diagrammformel möglich.

Beispiel:

```
Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/35.0.1916.114 Safari/537.36
```

ComputerName

Diese Funktion liefert den Namen des Computers, wie vom Betriebssystem angegeben. Die Verwendung ist sowohl im Datenladeskript als auch in der Diagrammformel möglich.

Wenn der Name des Computers mehr als 15 Zeichen lang ist, enthält der String nur die ersten 15 Zeichen.

```
ComputerName( )
```

DocumentName

Diese Funktion gibt einen String mit dem Namen der aktuellen Qlik Sense-App zurück – ohne Pfad, aber mit einer Erweiterung. Die Verwendung ist sowohl im Datenladeskript als auch in der Diagrammformel möglich.

```
DocumentName( )
```

DocumentPath

Diese Funktion gibt einen String mit dem vollständigen Pfad zur aktuellen Qlik Sense-App zurück. Die Verwendung ist sowohl im Datenladeskript als auch in der Diagrammformel möglich.

```
DocumentPath( )
```

Diese Funktion wird im Standardmodus nicht unterstützt.

DocumentTitle

Diese Funktion gibt einen String mit dem Titel der aktuellen Qlik Sense-App zurück. Die Verwendung ist sowohl im Datenladeskript als auch in der Diagrammformel möglich.

```
DocumentTitle( )
```

EngineVersion

Diese Funktion gibt die vollständige Versionsnummer der Qlik Sense-Engine als String zurück.

```
EngineVersion ()
```

GetCollationLocale

Diese Skriptfunktion liefert den Kulturnamen des Sortierungs-Gebietsschemas, das verwendet wird. Wenn die Variable CollationLocale nicht festgelegt wurde, wird das tatsächliche Gebietsschema des Benutzerrechners geliefert.

```
GetCollationLocale( )
```

GetObjectField

GetObjectField() gibt den Namen der Dimension zurück. **Index** ist eine optionale Ganzzahl, die die Dimension angibt, die zurückgegeben werden sollte.

```
GetObjectField - Diagrammfunktion([index])
```

GetRegistryString

Diese Funktion liefert den Wert eines Schlüssels in der Windows-Registrierung. Die Verwendung ist sowohl im Datenladeskript als auch in der Diagrammformel möglich.

```
GetRegistryString(path, key)
```

Diese Funktion wird im Standardmodus nicht unterstützt.

GetSysAttr

Mit dieser Funktion werden die Mandanten- und Bereichsdomänenattribute für die ausgewählte App zurückgegeben. Die Verwendung ist sowohl im Datenladeskript als auch in der Diagrammformel möglich.

```
GetSysAttr (name)
```

Wenn Sie diese Funktion in Qlik Sense Client-Managed verwenden, werden nur leere Datenwerte zurückgegeben.

IsPartialReload

Diese Funktion liefert -1 (True) bei der partiellen Ausführung des Skripts, anderenfalls 0 (False).

```
IsPartialReload ()
```

InObject

Die Diagrammfunktion **InObject()** wertet aus, ob das aktuelle Objekt innerhalb eines anderen Objekts mit der im Funktionsargument angegebenen ID enthalten ist oder nicht. Das Objekt kann ein Arbeitsblatt oder eine Visualisierung sein.

```
InObject - Diagrammfunktion(id_str)
```

ObjectId

Die Diagrammfunktion **ObjectId()** gibt die ID des Objekts zurück, in dem die Formel ausgewertet wird. Die Funktion übernimmt ein optionales Argument, das angibt, welcher Typ von Objekt die Funktion betrifft. Das Objekt kann ein Arbeitsblatt oder eine Visualisierung sein. Diese Funktion ist nur in Diagrammformeln verfügbar.

```
ObjectId - Diagrammfunktion([object_type_str])
```

OSUser

Diese Funktion liefert den Namen des Benutzers, der aktuell verbunden ist. Die Verwendung ist sowohl im Datenladeskript als auch in der Diagrammformel möglich.

```
OSUser( )
```

In Qlik Sense Desktop und Qlik Sense Client-Managed Mobile liefert diese Funktion immer 'Personal\Me'.

ProductVersion

Diese Funktion gibt die vollständige Qlik Sense-Version und -Buildnummer als String zurück.

Diese Funktion ist veraltet und wurde durch **EngineVersion()** ersetzt.

```
ProductVersion ()
```

ReloadTime

Diese Funktion liefert den Endzeitpunkt des letzten Datenladevorgangs. Die Verwendung ist sowohl im Datenladeskript als auch in der Diagrammformel möglich.

```
ReloadTime( )
```

StateName

StateName() gibt den Namen des alternativen Zustands der Visualisierung zurück, in der er verwendet wird. Mit StateName können Sie beispielsweise Visualisierungen mit dynamischem Text und dynamischen Farben erstellen, die eventuelle Änderungen an der Visualisierung angeben. Diese Funktion kann in Diagrammformeln verwendet werden, allerdings nicht, um den Status einer bezogenen Formel zu bestimmen.

```
StateName - Diagrammfunktion()
```

EngineVersion

Diese Funktion gibt die vollständige Versionsnummer der Qlik Sense-Engine als String zurück.

Syntax:

```
EngineVersion()
```

GetSysAttr

Mit dieser Funktion werden die Mandanten- und Bereichsdomänenattribute für die ausgewählte App zurückgegeben. Die Verwendung ist sowohl im Datenladeskript als auch in der Diagrammformel möglich.

Wenn Sie diese Funktion in Qlik Sense Client-Managed verwenden, werden leere Datenwerte zurückgegeben. Daher können Sie die Funktion zum Entwickeln von Ladeskripten in Qlik Sense Client-Managed verwenden, ohne dass Fehler auftreten, damit die Apps später an Qlik Cloud hochgeladen werden können.

Die vollständige Dokumentation für die Funktion Qlik Cloud finden Sie unter GetSysAttr – Skript- und Diagrammfunktion.

InObject - Diagrammfunktion

Die Diagrammfunktion **InObject()** wertet aus, ob das aktuelle Objekt innerhalb eines anderen Objekts mit der im Funktionsargument angegebenen ID enthalten ist oder nicht. Das Objekt kann ein Arbeitsblatt oder eine Visualisierung sein.

Diese Funktion kann verwendet werden, um die Hierarchie von Objekten in einem Arbeitsblatt zu zeigen, vom Arbeitsblattobjekt der obersten Ebene bis zu Visualisierungen, die in anderen Visualisierungen verschachtelt sind. Die Funktion kann zusammen mit den Funktionen **if** und **ObjectId** verwendet werden, um benutzerdefinierte Navigation in Ihren Apps zu erstellen.

Syntax:

```
InObject(id_str)
```

Rückgabe Datentyp: Boolesch

In Qlik Sense wird der boolesche Wert „wahr“ durch -1 dargestellt, der Wert „falsch“ durch 0.

Argumente

Argument	Beschreibung
id_str	Ein Stringwert, der die ID des auszuwertenden Objekts darstellt.

Die Arbeitsblatt-ID kann aus der App-URL abgerufen werden. Verwenden Sie für Visualisierungen die **Entwickler**-Optionen, um die Objekt-ID und den Textstring des Objekttyps zu identifizieren.

Gehen Sie folgendermaßen vor:

1. Fügen Sie im Analysemodus den folgenden Text zu Ihrer URL hinzu:
   /options/developer
2. Klicken Sie mit der rechten Maustaste auf eine Visualisierung und klicken Sie auf **Entwickler**.
3. Rufen Sie unter **Eigenschaften** die Objekt-ID aus der Dialogkopfzeile und den Objekttyp aus der Eigenschaft **"qType"** ab.

Beschränkungen:

Diese Funktion kann zu unerwarteten Ergebnissen führen, wenn sie in einem Objekt (z. B. einer Schaltfläche) innerhalb einer Sammelbox aufgerufen wird, die ein Master-Element ist. Diese Einschränkung gilt auch für Filterfenster-Master-Elemente, bei denen es sich um Sammelboxen für eine Reihe von Listboxen handelt. Das liegt daran, wie Master-Elemente die Objekthierarchie verwenden.

InObject() wird oft in Kombination mit den folgenden Funktionen verwendet:

Verwandte Funktionen

Funktion	Interaktion
if (page 583)	Die Funktionen **if** und **ObjectId** können zusammen verwendet werden, um bedingte Formeln zu erstellen. Beispielsweise können Visualisierungen bedingte Farben über Formeln erhalten, wenn diese Funktionen verwendet werden.
ObjectId - Diagrammfunktion (page 1516)	Ähnlich wie **if** wird auch **ObjectId** zusammen mit **InObject** verwendet, um bedingte Formeln zu erstellen.

Beispiel 1 – Grundlegende Funktionalität

Diagrammformel und Ergebnisse

Das folgende grundlegende Beispiel zeigt, wie Sie festlegen, ob ein Objekt innerhalb eines anderen Objekts enthalten ist. In diesem Fall prüfen wir, ob ein **Text und Bild**-Objekt sich in einem Arbeitsblattobjekt befindet, indem wir die ID des Arbeitsblatts als Argument verwenden.

Gehen Sie folgendermaßen vor:

1. Öffnen Sie ein neues Arbeitsblatt und ziehen Sie ein **Text und Bild**-Diagramm auf das Arbeitsblatt.
2. Klicken Sie im Eigenschaftsfenster auf **Kennzahl hinzufügen**.
3. Klicken Sie zum Öffnen des Formel-Editors auf fx.
4. Fügen Sie folgende Formel in das Dialogfeld ein:
   `=InObject()`
5. Ändern Sie die Formel, um die ID Ihres Arbeitsblatts als String zwischen den Klammern einzuschließen. Beispiel: Für ein Arbeitsblatt mit ID 1234-5678 verwenden Sie die folgende Formel:

```
=InObject('1234-5678')
```

6. Klicken Sie auf **Übernehmen**.

Der Wert -1 wird im Diagramm angezeigt und gibt an, dass die Formel als wahr ausgewertet wurde.

Beispiel 2 – Objekte mit bedingten Farben

Diagrammformel und Ergebnisse

Übersicht

Das folgende Beispiel zeigt, wie Sie benutzerdefinierte Navigationsschaltflächen mit verschiedenen Farben erstellen, um das derzeit geöffnete Arbeitsblatt anzugeben.

Erstellen Sie zuerst eine neue App und öffnen Sie den Dateneditor. Fügen Sie das folgende Ladeskript in eine neue Registerkarte ein: Beachten Sie, dass es sich bei den Daten selbst um einen Platzhalter handelt und sie im Beispielinhalt nicht verwendet werden.

Ladeskript

```
Transactions:

Load
*
Inline
[
id,date,amount
8188,'1/19/2022',37.23
8189,'1/7/2022',17.17
8190,'2/28/2022',88.27
8191,'2/5/2022',57.42
8192,'3/16/2022',53.80
8193,'4/1/2022',82.06
8194,'4/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'7/26/2022',45.89
8198,'8/9/2022',36.23
8199,'9/22/2022',25.66
8200,'11/23/2022',82.77
8201,'12/27/2022',69.98
8202,'1/1/2023',76.11
8203,'2/8/2022',25.12
8204,'3/19/2022',46.23
8205,'6/26/2022',84.21
8206,'9/14/2022',96.24
8207,'11/29/2022',67.67
];
```

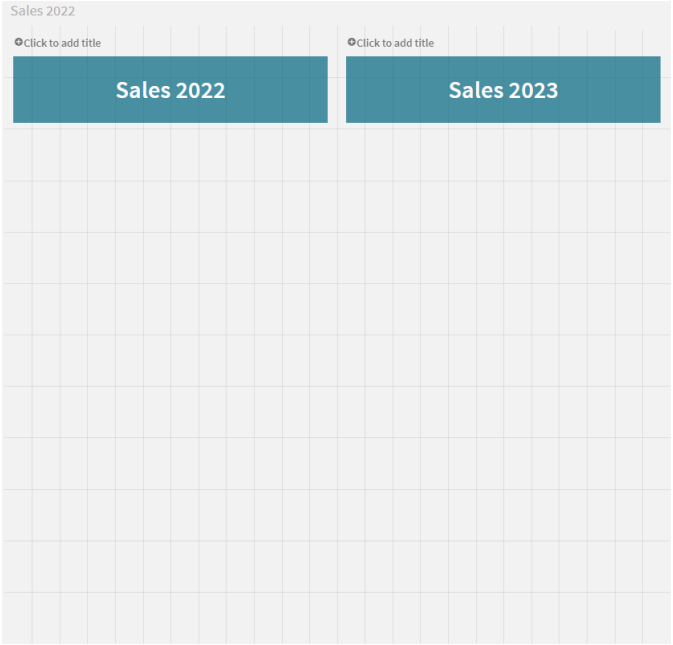
Erstellen der Visualisierungen

Laden Sie die Daten und erstellen Sie zwei neue Arbeitsblätter. Nennen Sie sie *Umsatz 2022* und *Umsatz 2023*.

Erstellen Sie dann zwei Schaltflächenobjekte, die zum Navigieren zwischen den beiden Arbeitsblättern verwendet werden.

Gehen Sie folgendermaßen vor:

1. Fügen Sie dem Arbeitsblatt zwei **Schaltfläche**-Objekte hinzu.
2. Legen Sie unter **Darstellung** > **Allgemein** die **Bezeichnung** der einzelnen Schaltflächen auf *Umsatz 2022* bzw. auf *Umsatz 2023* fest.
3. Ordnen Sie die Schaltflächen so an, dass sie der folgenden Abbildung entsprechen.
   Anordnung des Arbeitsblatts Umsatz 2022 mit zwei Navigationsschaltflächen



4. Wählen Sie die Schaltfläche *Umsatz 2022* aus und erweitern Sie im Eigenschaftsfenster **Aktionen und Navigation**.
5. Klicken Sie auf **Aktion hinzufügen** und wählen Sie unter **Navigation** die Option **Gehe zu Arbeitsblatt** aus.
6. Wählen Sie unter **Arbeitsblatt** die Option *Umsatz 2022* aus.
7. Wiederholen Sie diese Einrichtung der Schaltflächenaktion, um die Schaltfläche **Umsatz 2023** mit dem Arbeitsblatt *Umsatz 2023* zu verknüpfen.
8. Konvertieren Sie die Schaltflächen in Master-Elemente, indem Sie mit der rechten Maustaste daraufklicken und **Zu Master-Elementen hinzufügen** auswählen.

Jetzt können Sie jede Schaltfläche kopieren und in das Arbeitsblatt *Umsatz 2023* einfügen. Verwenden Sie dabei die gleiche Größe und Anordnung auf dem Arbeitsblatt.

Erstellen von bedingten Farben

Konfigurieren Sie dann die Schaltflächen so, dass sie blau angezeigt werden, wenn sie mit dem aktuell geöffneten Arbeitsblatt verknüpft sind, bzw. hellgrau, wenn sie mit dem nicht geöffneten Arbeitsblatt verknüpft sind.

Gehen Sie folgendermaßen vor:

1. Öffnen Sie das Arbeitsblatt *Umsatz 2022* und rufen Sie die Arbeitsblatt-ID aus der URL ab. Halten Sie das Arbeitsblatt *Umsatz 2022* geöffnet.
2. Klicken Sie auf das Schaltflächen-Master-Element **Umsatz 2022** und wählen Sie im Eigenschaftsfenster **Bearbeiten** aus.
3. Wählen Sie unter **Darstellung** > **Hintergrund** zum Einfärben der Schaltfläche die Option **Nach Formel** aus.
4. Fügen Sie unter **Formel** den folgenden Text ein:
   =if(InObject(''), Blue(), LightGray())
5. Fügen Sie zwischen den Klammern in der obigen Formel die Arbeitsblatt-ID für das Arbeitsblatt *Umsatz 2022* ein.

Die Schaltfläche ist jetzt so konfiguriert, dass sie blau wird, wenn das Arbeitsblatt *Umsatz 2022* geöffnet ist, bzw. hellgrau, wenn es nicht geöffnet ist.

Wiederholen Sie die obigen Anweisungen für das Arbeitsblatt *Umsatz 2023* und verknüpfen Sie dabei das Schaltflächen-Master-Element **Umsatz 2023** mit der Arbeitsblatt-ID für *Umsatz 2023*.

Jedes Arbeitsblatt hat jetzt zwei Schaltflächen, bei denen das aktuell geöffnete Arbeitsblatt mit der Farbe Blau angezeigt wird.

Arbeitsblatt Umsatz 2022 mit blauer Farbe, die angibt, dass „Umsatz 2022“ derzeit angezeigt wird

IsPartialReload

Diese Funktion liefert -1 (True) bei der partiellen Ausführung des Skripts, anderenfalls 0 (False).

Syntax:

```
IsPartialReload()
```

ObjectId - Diagrammfunktion

Die Diagrammfunktion **ObjectId()** gibt die ID des Objekts zurück, in dem die Formel ausgewertet wird. Die Funktion übernimmt ein optionales Argument, das angibt, welcher Typ von Objekt die Funktion betrifft. Das Objekt kann ein Arbeitsblatt oder eine Visualisierung sein. Diese Funktion ist nur in Diagrammformeln verfügbar.

Syntax:

```
ObjectId([object_type_str])
```

Rückgabe Datentyp: String

Das einzige Argument der Funktion, **object_type_str**, ist optional und verweist auf einen Stringwert, der den Objekttyp darstellt.

Argumente

Argument	Beschreibung
object_type_str	Ein Stringwert, der den Typ des auszuwertenden Objekts darstellt.

Wenn in der Funktionsformel kein Argument angegeben ist, gibt **ObjectId()** die ID des Objekts zurück, in dem die Formel verwendet wird. Um die ID des Arbeitsblattobjekts zurückzugeben, in dem die Visualisierung angezeigt wird, verwenden Sie *ObjectId('sheet')*.

Im Fall von Visualisierungsobjekten, die in anderen Visualisierungsobjekten verschachtelt sind, geben Sie den gewünschten Objekttyp im Funktionsargument an, um andere Ergebnisse zu erhalten. Verwenden Sie beispielsweise für ein **Text und Bild**-Diagramm in einer Sammelbox *'text-image'*, um das **Text und Bild**-Objekt zurückzugeben, und *'container'*, um die ID der Sammelbox zurückzugeben.

Gehen Sie folgendermaßen vor:

1. Fügen Sie im Analysemodus den folgenden Text zu Ihrer URL hinzu:
   /options/developer
2. Klicken Sie mit der rechten Maustaste auf eine Visualisierung und klicken Sie auf **Entwickler**.
3. Rufen Sie unter **Eigenschaften** die Objekt-ID aus der Dialogkopfzeile und den Objekttyp aus der Eigenschaft **"qType"** ab.

Beschränkungen:

Diese Funktion kann zu unerwarteten Ergebnissen führen, wenn sie in einem Objekt (z. B. einer Schaltfläche) innerhalb einer Sammelbox aufgerufen wird, die ein Master-Element ist. Diese Einschränkung gilt auch für Filterfenster-Master-Elemente, bei denen es sich um Sammelboxen für eine Reihe von Listboxen handelt. Das liegt daran, wie Master-Elemente die Objekthierarchie verwenden.

Die Diagrammformel *ObjectId('sheet')* gibt in diesen Fällen einen leeren String zurück, während *ObjectId ('masterobject')* die Identifikatoren des besitzenden Master-Elements anzeigt.

ObjectId() wird oft in Kombination mit den folgenden Funktionen verwendet:

Verwandte Funktionen

Funktion	Interaktion
if (page 583)	Die Funktionen **if** und **ObjectId** können zusammen verwendet werden, um bedingte Formeln zu erstellen. Beispielsweise können Visualisierungen bedingte Farben über Formeln erhalten, wenn diese Funktionen verwendet werden.
InObject - Diagrammfunktion (page 1511)	Ähnlich wie **if** wird auch **InObject** zusammen mit **ObjectId** verwendet, um bedingte Formeln zu erstellen.

Beispiel 1 – Zurückgeben der Diagrammobjekt-ID

Diagrammformel und Ergebnisse

Das folgende grundlegende Beispiel zeigt, wie die ID einer Visualisierung zurückgegeben wird.

Gehen Sie folgendermaßen vor:

1. Öffnen Sie ein neues Arbeitsblatt und ziehen Sie ein **Text und Bild**-Diagramm auf das Arbeitsblatt.
2. Klicken Sie im Eigenschaftsfenster auf **Kennzahl hinzufügen**.
3. Klicken Sie zum Öffnen des Formel-Editors auf fx.
4. Fügen Sie folgende Formel in das Dialogfeld ein:
   `=ObjectId()`
5. Klicken Sie auf **Übernehmen**.

Die ID des **Text und Bild**-Objekts wird in der Visualisierung angezeigt.

Das gleiche Ergebnis kann mit der folgenden Formel erzielt werden:

```
=ObjectId('text-image')
```

Beispiel 2 – Zurückgeben der Arbeitsblatt-ID

Diagrammformel und Ergebnisse
Das folgende grundlegende Beispiel zeigt, wie die ID des Arbeitsblatts zurückgegeben wird, auf dem eine Visualisierung angezeigt wird.

Gehen Sie folgendermaßen vor:

1. Öffnen Sie ein neues Arbeitsblatt und ziehen Sie ein **Text und Bild**-Diagramm auf das Arbeitsblatt.
2. Klicken Sie im Eigenschaftsfenster auf **Kennzahl hinzufügen**.
3. Klicken Sie zum Öffnen des Formel-Editors auf fx.
4. Fügen Sie folgende Formel in das Dialogfeld ein:
   `=ObjectId('sheet')`
5. Klicken Sie auf **Übernehmen**.

Die ID des Arbeitsblatts wird in der Visualisierung angezeigt.

Beispiel 3 – Verschachtelte Formel

Diagrammformel und Ergebnisse
Das folgende Beispiel zeigt, wie die Funktion **ObjectId()** innerhalb anderer Formeln verschachtelt werden kann.

Gehen Sie folgendermaßen vor:

1. Öffnen Sie ein neues Arbeitsblatt und ziehen Sie ein **Text und Bild**-Diagramm auf das Arbeitsblatt.
2. Klicken Sie im Eigenschaftsfenster auf **Kennzahl hinzufügen**.
3. Klicken Sie zum Öffnen des Formel-Editors auf fx.
4. Fügen Sie folgende Formel in das Dialogfeld ein:
   =if(InObject(ObjectId('text-image')), 'In Text und Bild', 'Nicht in Text und Bild')
5. Klicken Sie auf **Übernehmen**.

Der Text *In Text und Bild* wird im Diagramm angezeigt und gibt an, dass das in der Formel referenzierte Objekt ein **Text und Bild**-Diagramm ist.

Ein detaillierteres Beispiel mit bedingten Farben finden Sie unter *InObject - Diagrammfunktion (page 1511)*.

ProductVersion

Diese Funktion gibt die vollständige Qlik Sense-Version und -Buildnummer als String zurück. Diese Funktion ist veraltet und wurde durch **EngineVersion()** ersetzt.

Syntax:

```
ProductVersion()
```

StateName - Diagrammfunktion

StateName() gibt den Namen des alternativen Zustands der Visualisierung zurück, in der er verwendet wird. Mit StateName können Sie beispielsweise Visualisierungen mit dynamischem Text und dynamischen Farben erstellen, die eventuelle Änderungen an der Visualisierung angeben. Diese Funktion kann in Diagrammformeln verwendet werden, allerdings nicht, um den Status einer bezogenen Formel zu bestimmen.

Syntax:

```
StateName ()
```

Example 1:

```
      Dynamischer Text
='Region - ' & if(StateName() = '$', 'Default', StateName())
```

Example 2:

```
      Dynamische Farben
if(StateName() = 'Group 1', rgb(152, 171, 206),
      if(StateName() = 'Group 2', rgb(187, 200, 179),
            rgb(210, 210, 210)
      )
)
```

8.26 Tabellenfunktionen

Die Tabellenfunktionen liefern Informationen über die Tabelle, aus der Daten geladen werden. Ist kein Tabellenname angegeben, bezieht sich die Funktion auf die im jeweiligen **LOAD**-Befehl geladene Tabelle.

Im Datenladeskript können alle Funktionen verwendet werden. In einer Diagrammformel kann nur **NoOfRows** verwendet werden.

Tabellenfunktionen – Übersicht

Einige Funktionen werden nach der Übersicht genauer beschrieben. Bei diesen Funktionen können Sie auf den Funktionsnamen in der Syntax klicken, um direkt auf die Details zu der spezifischen Funktion zuzugreifen.

FieldName

Die Skriptfunktion **FieldName** liefert den Namen des Feldes der bereits geladenen Tabelle. Wird die Funktion in einem **LOAD**-Befehl verwendet, darf sie sich nicht auf die durch diesen Befehl entstehende Tabelle beziehen.

```
FieldName (field_number ,table_name)
```

FieldNumber
Die Skriptfunktion **FieldNumber** liefert die Nummer des Feldes in der bereits geladenen Tabelle. Wird die Funktion in einem **LOAD**-Befehl verwendet, darf sie sich nicht auf die durch diesen Befehl entstehende Tabelle beziehen.

```
FieldNumber (field_name ,table_name)
```

NoOfFields
Die Skriptfunktion **NoOfFields** liefert die Zahl der Felder in einer bereits geladenen Tabelle. Wird die Funktion in einem **LOAD**-Befehl verwendet, darf sie sich nicht auf die durch diesen Befehl entstehende Tabelle beziehen.

```
NoOfFields (table_name)
```

NoOfRows
Die Funktion **NoOfRows** liefert die Zahl der Zeilen (Datensätze) einer bereits geladenen Tabelle. Wird die Funktion in einem **LOAD**-Befehl verwendet, darf sie sich nicht auf die durch diesen Befehl entstehende Tabelle beziehen.

```
NoOfRows (table_name)
```

NoOfTables
Diese Skriptfunktion liefert die Zahl der bereits geladenen Tabellen.

```
NoOfTables()
```

TableName
Diese Skriptfunktion liefert den Namen der Tabelle mit der angegebenen Tabellennummer.

```
TableName(table_number)
```

TableNumber
Diese Skriptfunktion liefert die Nummer der angegebenen Tabelle. Die erste Tabelle trägt die Nummer 0.

Ist table_name nicht vorhanden, liefert diese Funktion NULL.

```
TableNumber(table_name)
```

Beispiel:

In diesem Beispiel möchten wir eine Tabelle mit Informationen über die geladenen Tabellen und Felder erstellen.

Zunächst laden wir Beispieldaten. Dadurch werden zwei Tabellen erstellt, mit deren Hilfe die in diesem Abschnitt beschriebenen Tabellenfunktionen näher erläutert werden sollen.

```
Characters:
Load Chr(RecNo()+Ord('A')-1) as Alpha, RecNo() as Num autogenerate 26;

ASCII:
Load
 if(RecNo()>=65 and RecNo()<=90,RecNo()-64) as Num,
 Chr(RecNo()) as AsciiAlpha,
```

```
 RecNo() as AsciiNum
autogenerate 255
 Where (RecNo()>=32 and RecNo()<=126) or RecNo()>=160 ;
```

Als Nächstes aggregieren wir durch die geladenen Tabellen mithilfe der Funktion **NoOfTables** und anschließend durch die Felder jeder Tabelle mithilfe der Funktion **NoOfFields** und laden Daten mithilfe der Tabellenfunktionen.

```
//Iterate through the loaded tables
For t = 0 to NoOfTables() - 1

//Iterate through the fields of table
 For f = 1 to NoOfFields(TableName($(t)))
  Tables:
  Load
   TableName($(t)) as Table,
   TableNumber(TableName($(t))) as TableNo,
   NoOfRows(TableName($(t))) as TableRows,
   FieldName($(f),TableName($(t))) as Field,
   FieldNumber(FieldName($(f),TableName($(t))),TableName($(t))) as FieldNo
   Autogenerate 1;
 Next f
Next t;
```

Die sich ergebende Tabelle Tables sieht folgendermaßen aus:

Load table

Table	TableNo	TableRows	Field	FieldNo
Characters	0	26	Alpha	1
Characters	0	26	Num	2
ASCII	1	191	Num	1
ASCII	1	191	AsciiAlpha	2
ASCII	1	191	AsciiNum	3

FieldName

Die Skriptfunktion **FieldName** liefert den Namen des Feldes der bereits geladenen Tabelle. Wird die Funktion in einem **LOAD**-Befehl verwendet, darf sie sich nicht auf die durch diesen Befehl entstehende Tabelle beziehen.

Syntax:

```
FieldName(field_number ,table_name)
```

Argumente:

Argumente

Argument	Beschreibung
field_number	Die Feldnummer des Felds, das Sie referenzieren möchten.
table_name	Die Tabelle mit dem Feld, das Sie referenzieren möchten.

Beispiel:

```
LET a = FieldName(4,'tab1');
```

FieldNumber

Die Skriptfunktion **FieldNumber** liefert die Nummer des Feldes in der bereits geladenen Tabelle. Wird die Funktion in einem **LOAD**-Befehl verwendet, darf sie sich nicht auf die durch diesen Befehl entstehende Tabelle beziehen.

Syntax:

```
FieldNumber(field_name ,table_name)
```

Argumente:

Argumente

Argument	Beschreibung
field_name	Name des Feldes.
table_name	Der Name der Tabelle, die das Feld enthält.

Falls das Feld field_name nicht in table_name vorhanden ist oder table_name nicht vorhanden ist, liefert die Funktion 0.

Beispiel:

```
LET a = FieldNumber('Customer','tab1');
```

NoOfFields

Die Skriptfunktion **NoOfFields** liefert die Zahl der Felder in einer bereits geladenen Tabelle. Wird die Funktion in einem **LOAD**-Befehl verwendet, darf sie sich nicht auf die durch diesen Befehl entstehende Tabelle beziehen.

Syntax:

```
NoOfFields(table_name)
```

Argumente:

Argumente

Argument	Beschreibung
table_name	Der Name der Tabelle.

Beispiel:

```
LET a = NoOfFields('tab1');
```

NoOfRows

Die Funktion **NoOfRows** liefert die Zahl der Zeilen (Datensätze) einer bereits geladenen Tabelle. Wird die Funktion in einem **LOAD**-Befehl verwendet, darf sie sich nicht auf die durch diesen Befehl entstehende Tabelle beziehen.

Syntax:

```
NoOfRows(table_name)
```

Argumente:

Argumente

Argument	Beschreibung
table_name	Der Name der Tabelle.

Beispiel:

```
LET a = NoOfRows('tab1');
```

8.27 Trigonometrische und hyperbolische Funktionen

Dieser Abschnitt beschreibt Funktionen für die Durchführung von trigonometrischen und hyperbolischen Funktionen. Bei allen Funktionen sind die Argumente Formeln, die im Bogenmaß gemessene Winkel ergeben, wobei **x** als reelle Zahl interpretiert werden sollte.

Alle Winkel sind im Bogenmaß anzugeben.

Alle Funktionen können sowohl im Datenladeskript als auch in den Diagrammformeln verwendet werden.

cos

Cosinus von **x**. Ergebnis ist eine Zahl zwischen -1 und 1.

```
cos( x )
```

acos

Inverser Cosinus von **x**. Diese Funktion ist nur für -1≤**x**≤1 definiert. Ergebnis ist eine Zahl zwischen 0 und π.

```
acos( x )
```

sin
Sinus von **x**. Ergebnis ist eine Zahl zwischen -1 und 1.

```
sin( x )
```

asin
Inverser Sinus von **x**. Diese Funktion ist nur für -1≤**x**≤1 definiert. Ergebnis ist eine Zahl zwischen -π/2 und π/2.

```
asin( x )
```

tan
Tangens von **x**. Das Ergebnis ist eine reelle Zahl.

```
tan( x )
```

atan
Inverser Tangens von **x**. Ergebnis ist eine Zahl zwischen -π/2 und π/2.

```
atan( x )
```

atan2
Zweidimensionale Verallgemeinerung der inversen Tangens-Funktion. Ergebnis ist der Winkel zwischen dem Ursprung und dem durch **x** und **y** definierten Punkt. Ergebnis ist eine Zahl zwischen - π and + π.

```
atan2( y,x )
```

cosh
Cosinus Hyberbolicus von **x**. Das Ergebnis ist eine positive reelle Zahl.

```
cosh( x )
```

sinh
Sinus Hyberbolicus von **x**. Das Ergebnis ist eine reelle Zahl.

```
sinh( x )
```

tanh
Tangens Hyberbolicus von **x**. Das Ergebnis ist eine reelle Zahl.

```
tanh( x )
```

acosh
Umgekehrter Cosinus Hyberbolicus von **x**. Das Ergebnis ist eine positive reelle Zahl.

```
acosh( x )
```

asinh
Umgekehrter Sinus Hyberbolicus von **x**. Das Ergebnis ist eine reelle Zahl.

```
asinh( x )
```

atanh
Umgekehrter Tangens Hyberbolicus von **x**. Das Ergebnis ist eine reelle Zahl.

```
atanh( x )
```

Beispiele:

Das folgende Skript lädt eine Beispieltabelle und anschließend eine Tabelle, welche die berechneten trigonometrischen und hyperbolischen Funktionen für die Werte enthält.

```
SampleData:
LOAD * Inline
[Value
-1
0
1];

Results:
Load *,
cos(Value),
acos(Value),
sin(Value),
asin(Value),
tan(Value),
atan(Value),
atan2(Value, Value),
cosh(Value),
sinh(Value),
tanh(Value)
RESIDENT SampleData;

Drop Table SampleData;
```

8.28 Window-Funktionen

Window-Funktionen führen Berechnungen mit Werten aus mehreren Zeilen durch, um einen getrennten Wert für jede Zeile zurückzugeben. Window-Funktionen können erst berechnet werden, nachdem die ganze Tabelle gelesen wurde.

Sie können die Window-Funktionen für Vorgänge wie die Folgenden verwenden:

- Einen einzelnen Zahlenwert in einer Zeile mit dem Durchschnitt, Höchstwert oder Mindestwert in der Spalte vergleichen.
- Den Rang eines einzelnen Werts entweder in der Spalte oder in der ganzen Tabelle berechnen.

Window-Funktionen ändern die Anzahl der Datensätze in der Tabelle nicht, können jedoch ähnliche Aufgaben als Aggregierungs-, relationale und Bereichsfunktionen durchführen.

Jede Funktion wird nach der Übersicht genauer beschrieben. Sie können auch auf den Funktionsnamen in der Syntax klicken, um direkt auf die Details zu der spezifischen Funktion zuzugreifen.

Window
Die **Window**-Funktion führt Berechnungen für mehrere Zeilen durch und gibt einen getrennten Wert für jede Zeile zurück.

```
Window – Skriptfunktion(input_expr, [partition1, partition2, ...], [sort_
type, [sort_expr]],[filter_expr], [start_expr,end_expr])[row_window_size])
```

WRank
Die **WRank**-Funktion führt Rangfolgenberechnungen innerhalb von **Window** durch.

```
WRank – Skriptfunktion([TOTAL] expr[, mode[, fmt]])
```

Window – Skriptfunktion

Window() führt Berechnungen für mehrere Zeilen durch und gibt einen getrennten Wert für jede Zeile zurück.

Sie können die **Window**-Funktionen für Vorgänge wie die Folgenden verwenden:

- Einen einzelnen Zahlenwert in einer Zeile mit dem Durchschnitt, Höchstwert oder Mindestwert in der Spalte vergleichen.
- Den Rang eines einzelnen Werts entweder in der Spalte oder in der ganzen Tabelle berechnen.

Die **Window**-Funktion ändert nicht die Anzahl der Datensätze in der Tabelle, kann jedoch ähnliche Aufgaben als Aggregierung, relationale und Bereichsfunktionen durchführen.

Zum Hinzufügen zur Tabelle muss die **Window**-Funktion einen Cache innerhalb des LOAD-Befehls der Tabelle enthalten, mit der Sie arbeiten. Hier ein Beispiel:

```
[Transactions]:
Load
       *,
       Window(avg(Expression1),[Num]);
LOAD
       TransLineID,
       TransID,
       "Num",
       Dim1,
       Dim2,
       Dim3,
       Expression1,
       Expression2,
       Expression3
FROM [lib://AttachedFiles/transactions.qvd] (qvd);
```

„Window“ unterstützt allgemeine Funktionen wie Rundungen oder grundlegende numerische Operationen. Hier ein Beispiel:

```
Load *, Round(Window(Sum(Salary),Department)) as SumSalary
Load *, Window(Sum(Salary),Department) + 5 as SumSalary
```

Sie können ein Sliding Window für die **Window**-Funktion definieren. Damit wird die Anzahl der Zeilen festgelegt, die beim Anwenden der **Window**-Funktion auf die aktuelle Zeile verwendet werden. Beispielsweise können Sie das Fenster auf die 3 vorherigen und die 3 darauffolgenden Zeilen festlegen.

Syntax:

```
Window (input_expr, [partition1, partition2, ...], [sort_type, [sort_expr]],
[filter_expr], [start_expr,end_expr])
```

Rückgabedatentyp: Ein neues Feld, das der durch den LOAD-Befehl erstellten Tabelle hinzugefügt wurde.

Argumente:

Argumente

Argument	Beschreibung

input_expr	Die von der Funktion berechnete und zurückgegebene Eingabeformel. Es muss sich um eine beliebige, auf einer Aggregierung basierende Formel handeln, z. B. `Median(Salary)`. Hier ein Beispiel:
	`Window(Median(Salary)) as MedianSalary`
	Die Eingabe kann auch ein Feldname ohne angewendete Aggregierung sein. In diesem Fall behandelt **Window** ihn so, als würde die **Only()**-Funktion auf dieses Feld angewendet. Hier ein Beispiel:
	`Window(Salary,Department) as WSalary`
	
	Optional können Sie Partitionierung mit der Eingabeformel definieren. Partitionierung entspricht der Gruppierung, die mithilfe des **group by**-Befehls erzielt wird, mit dem Unterschied, dass das Ergebnis der Eingabetabelle als neue Spalte hinzugefügt wird. Partitionierung reduziert die Anzahl der Datensätze in der Eingabetabelle nicht. Es können mehrere Partitionsfelder definiert werden.
	Beispiel:
	`LOAD`
	`Window(Max(Sales), City, 'ASC', OrderDate, Sales > 300)`
	`+ AddMonths(OrderDate,-6) as MAX_Sales_City_Last_6_Mos,`
	`Window(Avg(Sales), City, 'ASC', OrderDate, City = 'Portland')`
	`+ AddMonths(OrderDate,-6) as Avg_Sales_Portland_Last_6_Mos,`
	`Window(Max(Sales), City, 'ASC', OrderDate, Sales > 300)`
	`+ AddMonths(OrderDate,-12) as MAX_Sales_City_Last_12_Mos;`
	`LOAD`
	`City,`
	`Sales,`
	`OrderDate`
	`FROM [lib://AttachedFiles/Sales Data.xlsx]`
	`(ooxml, embedded labels, table is [Sales Data]);`

Argument	Beschreibung
partition1, partition2	Nach input_expr können Sie eine beliebige Anzahl Partitionen definieren. Partitionen sind Felder, die definieren, auf welche Kombinationen die Aggregierungen angewendet werden sollen. Die Aggregierung wird für jede Partition getrennt angewendet. Hier ein Beispiel: `Window(Avg(Salary), Unit, Department, Country) as AvgSalary` Im obigen Beispiel sind die Partitionen *Unit*, *Department* und *Country*. Partitionen sind nicht obligatorisch, aber für ordnungsgemäßes Windowing von Feldern erforderlich.
sort_type, [sort_expr]]	Geben Sie optional den Sortiertyp und die Sortierformel an. sort_type kann einen von zwei Werten haben: • ASC: Aufsteigende Sortierung. • DESC: Absteigende Sortierung. Wenn Sie „sort_type“ definieren, müssen Sie eine Sortierformel definieren. Dies ist eine Formel, die die Zeilenreihenfolge innerhalb einer Partition festlegt. Hier ein Beispiel: `Window(RecNo(), Department, 'ASC', Year)` Im obigen Beispiel werden die Ergebnisse innerhalb der Partition aufsteigend vom Feld *Year* sortiert. *Der Sortiertyp und die Sortierformel sind im Wesentlichen nur für die Funktionen **RecNo** und **WRank** erforderlich.*
filter_expr	Fügen Sie optional eine Filterformel hinzu. Dies ist eine boolesche Formel, die festlegt, ob der Datensatz in die Berechnung einbezogen werden soll oder nicht. Dieser Parameter kann vollständig ausgelassen werden, und das Ergebnis gibt an, dass kein Filter vorhanden ist. Hier ein Beispiel: `Window(avg(Salary), Department, 'ASC', Age, EmployeeID=3 Or EmployeeID=7) as wAvgSalary) as wAvgSalaryIfEmpIs3or7`

Argument	Beschreibung

[start_expr,end_expr]	Legen Sie optional das Argument für die Sliding Window-Funktionalität fest. Ein Sliding Window erfordert zwei Argumente:
	• Startformel: Die Anzahl der Zeilen vor der aktuellen Zeile, die in das Fenster eingeschlossen werden sollen.
	• Endformel: Die Anzahl der Zeilen nach der aktuellen Zeile, die in das Fenster eingeschlossen werden sollen.
	Wenn Sie beispielsweise die 3 vorherigen Zeilen, die aktuelle Zeile und die nachfolgende Zeile einschließen möchten:
	`Window(concat(Text(Salary),'-'), Department, 'ASC', Age, Year>0,  -3, 1) as WSalaryDepartment`
	Um alle vorherigen oder alle nachfolgenden Zeilen anzugeben, können Sie die Funktion **Unbounded()** verwenden. Wenn Sie beispielsweise alle vorhergehenden Zeilen, die aktuelle Zeile und die nachfolgende Zeile einschließen möchten:
	`Window(concat(Text(Salary),'-'), Department, 'ASC', Age, Year>0,  UNBOUNDED(), 1) as WSlidingSalaryDepartment`
	Beispiel zum Einschließen der dritten Zeile ab der aktuellen Zeile und aller nachfolgenden Zeilen:
	`Window(concat(Text(Salary),'-'), Department, 'ASC', Age, Year>0,  3, UNBOUNDED()) as WSlidingSalaryDepartment`

Beispiel: Hinzufügen eines Felds, das eine Aggregierung enthält

Beispiel: Hinzufügen eines Felds, das eine Aggregierung enthält

Ladeskript

Erstellen Sie eine neue Registerkarte im Dateneditor und laden Sie die folgenden Daten als Inline-Ladevorgang: Erstellen Sie die Tabelle unten in Qlik Sense, um die Ergebnisse anzuzeigen.

```
Transactions:
Load
*,
Window(Avg(transaction_amount),customer_id) as AvgCustTransaction;

Load * Inline [
transaction_id, transaction_date, transaction_amount, transaction_quantity, customer_id, size, color_code
3750, 20180830, 23.56, 2, 2038593, L, Red
3751, 20180907, 556.31, 6, 203521, M, Orange
3752, 20180916, 5.75, 1, 5646471, S, Blue
3753, 20180922, 125.00, 7, 3036491, L, Black
```

```
3754, 20180922, 484.21, 13, 049681, XS, Red
3756, 20180922, 59.18, 2, 2038593, M, Blue
3757, 20180923, 177.42, 21, 203521, XL, Black
3758, 20180924, 153.42, 14, 2038593, L, Red
3759, 20180925, 7.42, 5, 203521, M, Orange
3760, 20180925, 80.12, 18, 5646471, M, Blue
3761, 20180926, 3.42, 7, 3036491, XS, Black
3763, 20180926, 63.55, 12, 049681, S, Red
3763, 20180927, 177.56, 10, 2038593, L, Blue
3764, 20180927, 325.95, 8, 203521, XL, Black
];
```

Ergebnisse

Ergebnisse für das Hinzufügen eines Felds, das eine Aggregierung enthält

transaction_id	transaction_date	transaction_amount	transaction_quantity	customer_id	size	color_code	AvgCustTransaction
3750	20180830	23.56	2	2038593	L	Rot	103.43
3751	20180907	556.31	6	203521	M	Orange	266.775
3752	20180916	5.75	1	5646471	S	Blau	42.935
3753	20180922	125.00	7	3036491	L	Schwarz	64.21
3754	20180922	484.21	13	049681	XS	Rot	273.88
3756	20180922	59.18	2	2038593	M	Blau	103.43
3757	20180923	177.42	21	203521	XL	Schwarz	266.775
3758	20180924	153.42	14	2038593	L	Rot	103.43
3759	20180925	7.42	5	203521	M	Orange	266.775
3760	20180925	80.12	18	5646471	M	Blau	42.935
3761	20180926	3.42	7	3036491	XS	Schwarz	64.21
3763	20180926	63.55	12	049681	S	Rot	273.88
3763	20180927	177.56	10	2038593	L	Blau	103.43
3764	20180927	325.95	8	203521	XL	Schwarz	266.775

Beispiel:– Hinzufügen eines Felds, das eine nach bestimmten Werten gefilterte Aggregierung enthält

Beispiel: Hinzufügen eines Felds, das eine nach bestimmten Werten gefilterte Aggregierung enthält

Ladeskript

Erstellen Sie eine neue Registerkarte im Dateneditor und laden Sie die folgenden Daten als Inline-Ladevorgang: Erstellen Sie die Tabelle unten in Qlik Sense, um die Ergebnisse anzuzeigen.

```
Transactions:
Load
*,
Window(Avg(transaction_amount),customer_id, color_code = 'Blue') as AvgCustTransaction;

Load * Inline [
transaction_id, transaction_date, transaction_amount, transaction_quantity, customer_id, size,
color_code
3750, 20180830, 23.56, 2, 2038593, L, Red
3751, 20180907, 556.31, 6, 203521, M, Orange
3752, 20180916, 5.75, 1, 5646471, S, Blue
3753, 20180922, 125.00, 7, 3036491, L, Black
3754, 20180922, 484.21, 13, 049681, XS, Red
3756, 20180922, 59.18, 2, 2038593, M, Blue
3757, 20180923, 177.42, 21, 203521, XL, Black
3758, 20180924, 153.42, 14, 2038593, L, Red
3759, 20180925, 7.42, 5, 203521, M, Orange
3760, 20180925, 80.12, 18, 5646471, M, Blue
3761, 20180926, 3.42, 7, 3036491, XS, Black
3763, 20180926, 63.55, 12, 049681, S, Red
3763, 20180927, 177.56, 10, 2038593, L, Blue
3764, 20180927, 325.95, 8, 203521, XL, Black
];
```

Ergebnisse

Ergebnisse für das Hinzufügen eines Felds, das eine nach bestimmten Werten gefilterte Aggregierung enthält

transaction_id	transaction_date	transaction_amount	transaction_quantity	customer_id	size	color_code	AvgCustTransaction
3750	20180830	23.56	2	2038593	L	Rot	-
3751	20180907	556.31	6	203521	M	Orange	-
3752	20180916	5.75	1	5646471	S	Blau	42.94
3753	20180922	125.00	7	3036491	L	Schwarz	-
3754	20180922	484.21	13	049681	XS	Rot	-

transaction_id	transaction_date	transaction_amount	transaction_quantity	customer_id	size	color_code	AvgCustTransaction
3756	20180922	59.18	2	2038593	M	Blau	118.4
3757	20180923	177.42	21	203521	XL	Schwarz	-
3758	20180924	153.42	14	2038593	L	Rot	-
3759	20180925	7.42	5	203521	M	Orange	-
3760	20180925	80.12	18	5646471	M	Blau	42.94
3761	20180926	3.42	7	3036491	XS	Schwarz	-
3763	20180926	63.55	12	049681	S	Rot	-
3763	20180927	177.56	10	2038593	L	Blau	118.4
3764	20180927	325.95	8	203521	XL	Schwarz	-

Beispiel: Hinzufügen eines Felds mit einem Sliding Window

Beispiel: Hinzufügen eines Felds mit einem Sliding Window

Ladeskript

Erstellen Sie eine neue Registerkarte im Dateneditor und laden Sie die folgenden Daten als Inline-Ladevorgang: Erstellen Sie die Tabelle unten in Qlik Sense, um die Ergebnisse anzuzeigen.

```
Transactions:
Load
*,
Window(Avg(transaction_amount),customer_id, 'ASC', -1, 1, 0, 1) as AvgCustTransaction;

Load * Inline [
transaction_id, transaction_date, transaction_amount, transaction_quantity, customer_id, size,
color_code
3750, 20180830, 23.56, 2, 2038593, L, Red
3751, 20180907, 556.31, 6, 203521, M, Orange
3752, 20180916, 5.75, 1, 5646471, S, Blue
3753, 20180922, 125.00, 7, 3036491, L, Black
3754, 20180922, 484.21, 13, 049681, XS, Red
3756, 20180922, 59.18, 2, 2038593, M, Blue
3757, 20180923, 177.42, 21, 203521, XL, Black
3758, 20180924, 153.42, 14, 2038593, L, Red
3759, 20180925, 7.42, 5, 203521, M, Orange
3760, 20180925, 80.12, 18, 5646471, M, Blue
3761, 20180926, 3.42, 7, 3036491, XS, Black
3763, 20180926, 63.55, 12, 049681, S, Red
3763, 20180927, 177.56, 10, 2038593, L, Blue
3764, 20180927, 325.95, 8, 203521, XL, Black
```

```
];
```

Ergebnisse

Ergebnisse für das Hinzufügen eines Felds, das eine nach bestimmten Werten gefilterte Aggregierung enthält

transaction_id	transaction_date	transaction_amount	transaction_quantity	customer_id	size	color_code	AvgCustTransaction
3750	20180830	23.56	2	2038593	L	Rot	41.37
3751	20180907	556.31	6	203521	M	Orange	366.865
3752	20180916	5.75	1	5646471	S	Blau	42.935
3753	20180922	125.00	7	3036491	L	Schwarz	64.21
3754	20180922	484.21	13	049681	XS	Rot	273.88
3756	20180922	59.18	2	2038593	M	Blau	106.3
3757	20180923	177.42	21	203521	XL	Schwarz	92.42
3758	20180924	153.42	14	2038593	L	Rot	165.49
3759	20180925	7.42	5	203521	M	Orange	166.685
3760	20180925	80.12	18	5646471	M	Blau	80.12
3761	20180926	3.42	7	3036491	XS	Schwarz	3.42
3763	20180926	63.55	12	049681	S	Rot	177.56
3763	20180927	177.56	10	2038593	L	Blau	63.55
3764	20180927	325.95	8	203521	XL	Schwarz	325.95

Beschränkungen

Window hat die folgenden Einschränkungen:

- **Window** ist eine ressourcenintensive Funktion, vor allem hinsichtlich der Arbeitsspeichernutzung.
- **Window** wird in Qlik Sense Mobile nicht unterstützt.
- Diagrammformeln unterstützen **Window** nicht.
- Sie können **Window**-Funktionen nicht in anderen **Window**-Funktionen verschachteln.
- **Window** kann nicht innerhalb einer Aggregierungsfunktion verwendet werden.
- **Window** muss die ganze Tabelle scannen können.
- **WRank()**, **RecNo()** und **RowNo()** können nicht mit **Window** verwendet werden, wenn die Sliding Window-Funktionalität verwendet wird.

WRank – Skriptfunktion

WRank() wertet die Zeilen einer Tabelle im Ladeskript aus und zeigt für jede Zeile die relative Position des Wertes des Felds an, das im Ladeskript ausgewertet wird. Beim Auswerten der Tabelle vergleicht die Funktion das Ergebnis mit den Ergebnissen der anderen Zeilen, die die aktuelle Partition enthalten, und liefert den Rang der aktuellen Zeile innerhalb des Segments.

Partitionen in einer Tabelle

Region	Country	Population	Rank(Population)
Americas	Mexico	128,932,753	2
Americas	Canada	37,742,154	3
Americas	United States of America	331,002,651	1
Europe	Sweden	10,099,265	4
Europe	United Kingdom	67,886,011	2
Europe	France	65,273,511	3
Europe	Germany	83,783,942	1

WRank kann nur in einer **Window**-Funktion verwendet werden. Die **Window**-Funktion muss einen Sortiertyp und eine Sortierformel enthalten. Die Rangfolge wird auf die Sortierformel angewendet.

Syntax:

```
WRank ([mode[, fmt]])
```

Rückgabe Datentyp: dual

Argumente:

Argumente

Argument	Beschreibung
mode	Gibt optional die numerische Darstellung des Funktionsergebnisses an.
fmt	Gibt optional die Textdarstellung des Funktionsergebnisses an.
TOTAL	Wenn die Tabelle nur eine Dimension hat oder dem Skript der Qualifizierer **TOTAL** vorangestellt ist, wird die Funktion über die gesamte Spalte ausgewertet. Hat die Tabelle oder Tabellenentsprechung mehrere vertikale Dimensionen, so umfasst die aktuelle Partition nur Zeilen, deren Werte in allen Dimensionsspalten mit Ausnahme der letzten Dimension in der Sortierfolge zwischen den Feldern mit der aktuellen Zeile übereinstimmen.

Der Rang ist eine duale Größe. Hat jede Spalte einen eindeutigen Rang, wird dieser als ganze Zahl zwischen 1 und der Zahl der Zeilen der aktuellen Partition ausgegeben.

Teilen sich mehrere Zeilen denselben Rang, kann das Ergebnis der Funktion durch die Parameter **mode** und **fmt** modifiziert werden.

mode

Das erste Argument **mode** kann folgende Werte annehmen:

mode-Werte

Wert	Beschreibung
0 (Standard)	Fallen die Ränge alle unterhalb des mittleren Ranges der gesamten Rangfolge, erhalten alle Zeilen der Gruppe den geringsten innerhalb dieser Gruppe auftretenden Rang. Fallen die Ränge alle oberhalb des mittleren Ranges der gesamten Rangfolge, erhalten alle Zeilen der Gruppe den höchsten innerhalb dieser Gruppe auftretenden Rang. Geht die Gruppe der Zeilen gleichen Rangs über den mittleren Rang der gesamten Rangfolge hinweg, erhalten alle Zeilen der Gruppe den mittleren Rang der gesamten Partition.
1	Alle Zeilen erhalten den niedrigsten Rang.
2	Alle Zeilen erhalten den mittleren Rang.
3	Alle Zeilen erhalten den höchsten Rang.
4	Die erste Zeile erhält den niedrigsten Rang, für die Ränge nachfolgender Zeilen wird jeweils eins addiert.

fmt

Das zweite Argument **fmt** kann folgende Werte annehmen:

fmt-Werte

Wert	Beschreibung
0 (Standard)	Niedrigster Wert - höchster Wert in allen Zeilen (z. B. 3 - 4).
1	Niedrigster Wert in allen Zeilen.
2	Niedrigster Wert in der ersten Zeile, leere Zeilen in allen weiteren Zeilen.

Die für **mode** 4 und **fmt** 2 maßgebliche Zeilenreihenfolge wird von der Ladereihenfolge der Tabellenfelder bestimmt.

Beispiel: Hinzufügen eines Felds mit Rangfolge

Beispiel: Hinzufügen eines Felds mit Rangfolge

Ladeskript

Erstellen Sie eine neue Registerkarte im Dateneditor und laden Sie die folgenden Daten als Inline-Ladevorgang: Erstellen Sie die Tabelle unten in Qlik Sense, um die Ergebnisse anzuzeigen.

```
Transactions:
Load
*,
Window(WRank(0),customer_id, 'Desc', transaction_amount) as TransactionRanking;
```

```
Load * Inline [
transaction_id, transaction_date, transaction_amount, transaction_quantity, customer_id, size,
color_code
3750, 20180830, 23.56, 2, 2038593, L, Red
3751, 20180907, 556.31, 6, 203521, M, Orange
3752, 20180916, 5.75, 1, 5646471, S, Blue
3753, 20180922, 125.00, 7, 3036491, L, Black
3754, 20180922, 484.21, 13, 049681, XS, Red
3756, 20180922, 59.18, 2, 2038593, M, Blue
3757, 20180923, 177.42, 21, 203521, XL, Black
3758, 20180924, 153.42, 14, 2038593, L, Red
3759, 20180925, 7.42, 5, 203521, M, Orange
3760, 20180925, 80.12, 18, 5646471, M, Blue
3761, 20180926, 3.42, 7, 3036491, XS, Black
3763, 20180926, 63.55, 12, 049681, S, Red
3763, 20180927, 177.56, 10, 2038593, L, Blue
3764, 20180927, 325.95, 8, 203521, XL, Black
];
```

Ergebnisse

Ergebnisse für das Hinzufügen eines Felds mit Rangfolge

transaction_id	transaction_date	transaction_amount	transaction_quantity	customer_id	size	color_code	TransactionRanking
3750	20180830	23.56	2	2038593	L	Rot	4-4
3751	20180907	556.31	6	203521	M	Orange	1-1
3752	20180916	5.75	1	5646471	S	Blau	2-2
3754	20180922	484.21	13	049681	XS	Rot	1-1
3756	20180922	59.18	2	2038593	M	Blau	3-3
3753	20180922	125.00	7	3036491	L	Schwarz	1-1
3757	20180923	177.42	21	203521	XL	Schwarz	3-3
3758	20180924	153.42	14	2038593	L	Rot	2-2
3759	20180925	7.42	5	203521	M	Orange	4-4
3760	20180925	80.12	18	5646471	M	Blau	1-1
3763	20180926	63.55	12	049681	S	Rot	2-2
3761	20180926	3.42	7	3036491	XS	Schwarz	2-2
3764	20180927	325.95	8	203521	XL	Schwarz	2-2
3763	20180927	177.56	10	2038593	L	Blau	1-1

Beispiel: Hinzufügen eines Felds mit Rangfolge mithilfe von fmt für ein einstelliges Ergebnis

Beispiel: Hinzufügen eines Felds mit Rangfolge mithilfe von fmt für ein einstelliges Ergebnis

Ladeskript

Erstellen Sie eine neue Registerkarte im Dateneditor und laden Sie die folgenden Daten als Inline-Ladevorgang: Erstellen Sie die Tabelle unten in Qlik Sense, um die Ergebnisse anzuzeigen.

```
Transactions:
Load
*,Window(WRank(0,1),customer_id, 'Desc', transaction_amount) as TransactionRanking;

Load * Inline [
transaction_id, transaction_date, transaction_amount, transaction_quantity, customer_id, size,
color_code
3750, 20180830, 23.56, 2, 2038593, L, Red
3751, 20180907, 556.31, 6, 203521, M, Orange
3752, 20180916, 5.75, 1, 5646471, S, Blue
3753, 20180922, 125.00, 7, 3036491, L, Black
3754, 20180922, 484.21, 13, 049681, XS, Red
3756, 20180922, 59.18, 2, 2038593, M, Blue
3757, 20180923, 177.42, 21, 203521, XL, Black
3758, 20180924, 153.42, 14, 2038593, L, Red
3759, 20180925, 7.42, 5, 203521, M, Orange
3760, 20180925, 80.12, 18, 5646471, M, Blue
3761, 20180926, 3.42, 7, 3036491, XS, Black
3763, 20180926, 63.55, 12, 049681, S, Red
3763, 20180927, 177.56, 10, 2038593, L, Blue
3764, 20180927, 325.95, 8, 203521, XL, Black
];
```

Ergebnisse

Ergebnisse für das Hinzufügen eines Felds mit Rangfolge mithilfe von fmt für ein einstelliges Ergebnis

transaction_id	transaction_date	transaction_amount	transaction_quantity	customer_id	size	color_code	TransactionRanking
3750	20180830	23.56	2	2038593	L	Rot	4
3751	20180907	556.31	6	203521	M	Orange	1
3752	20180916	5.75	1	5646471	S	Blau	2
3754	20180922	484.21	13	049681	XS	Rot	1
3756	20180922	59.18	2	2038593	M	Blau	3
3753	20180922	125.00	7	3036491	L	Schwarz	1

transaction_id	transaction_date	transaction_amount	transaction_quantity	customer_id	size	color_code	TransactionRanking
3757	20180923	177.42	21	203521	XL	Schwarz	3
3758	20180924	153.42	14	2038593	L	Rot	2
3759	20180925	7.42	5	203521	M	Orange	4
3760	20180925	80.12	18	5646471	M	Blau	1
3763	20180926	63.55	12	049681	S	Rot	2
3761	20180926	3.42	7	3036491	XS	Schwarz	2
3764	20180927	325.95	8	203521	XL	Schwarz	2
3763	20180927	177.56	10	2038593	L	Blau	1

Beispiel: Hinzufügen eines Felds mit Rangfolge mit mehreren Partitionen

Beispiel: Hinzufügen eines Felds mit Rangfolge mit mehreren Partitionen

Ladeskript

Erstellen Sie eine neue Registerkarte im Dateneditor und laden Sie die folgenden Daten als Inline-Ladevorgang: Erstellen Sie die Tabelle unten in Qlik Sense, um die Ergebnisse anzuzeigen.

```
Transactions:
Load
*,Window(WRank(0,1),customer_id, size, color_code, 'Desc', transaction_amount) as
TransactionRanking;

Load * Inline [
transaction_id, transaction_date, transaction_amount, transaction_quantity, customer_id, size,
color_code
3750, 20180830, 23.56, 2, 2038593, L, Red
3751, 20180907, 556.31, 6, 203521, M, Orange
3752, 20180916, 5.75, 1, 5646471, S, Blue
3753, 20180922, 125.00, 7, 3036491, L, Black
3754, 20180922, 484.21, 13, 049681, XS, Red
3756, 20180922, 59.18, 2, 2038593, M, Blue
3757, 20180923, 177.42, 21, 203521, XL, Black
3758, 20180924, 153.42, 14, 2038593, L, Red
3759, 20180925, 7.42, 5, 203521, M, Orange
3760, 20180925, 80.12, 18, 5646471, M, Blue
3761, 20180926, 3.42, 7, 3036491, XS, Black
3763, 20180926, 63.55, 12, 049681, S, Red
3763, 20180927, 177.56, 10, 2038593, L, Blue
3764, 20180927, 325.95, 8, 203521, XL, Black
];
```

Ergebnisse

Ergebnisse für das Hinzufügen eines Felds mit Rangfolge mithilfe von fmt für ein einstelliges Ergebnis

transaction_id	transaction_date	transaction_amount	transaction_quantity	customer_id	size	color_code	TransactionRanking
3750	20180830	23.56	2	2038593	L	Rot	2
3751	20180907	556.31	6	203521	M	Orange	1
3752	20180916	5.75	1	5646471	S	Blau	1
3754	20180922	484.21	13	049681	XS	Rot	1
3756	20180922	59.18	2	2038593	M	Blau	1
3753	20180922	125.00	7	3036491	L	Schwarz	1
3757	20180923	177.42	21	203521	XL	Schwarz	2
3758	20180924	153.42	14	2038593	L	Rot	1
3759	20180925	7.42	5	203521	M	Orange	2
3760	20180925	80.12	18	5646471	M	Blau	1
3763	20180926	63.55	12	049681	S	Rot	1
3761	20180926	3.42	7	3036491	XS	Schwarz	1
3764	20180927	325.95	8	203521	XL	Schwarz	1
3763	20180927	177.56	10	2038593	L	Blau	1

Beschränkungen

WRank weist die folgenden Beschränkungen auf:

- Wenn Ihr fmt-Wert 0 ist und Sie den Textteil des dualen Ergebnisses für **WRank** verwenden möchten, müssen Sie **Text()** mit **Window(WRank)** verwenden. Hier ein Beispiel: `Text(Window(WRank(0), Unit, 'DESC', Age)) as UnitWRankedByAgeText`.

9 Zugriffsbeschränkung für Dateisystem

Aus Sicherheitsgründen unterstützt Qlik Sense im Standardmodus keine Pfade im Datenladeskript oder in den Funktionen und Variablen, die das Dateisystem präsentieren.

Da Dateisystempfade jedoch in QlikView unterstützt wurden, können Sie den Standardmodus deaktivieren und den Legacymodus verwenden, um QlikView-Ladeskripte wiederverwenden zu können.

Die Deaktivierung des Standardmodus kann ein Sicherheitsrisiko darstellen, da das Dateisystem offengelegt wird.

Deaktivieren des Standardmodus (page 1547)

9.1 Sicherheitsaspekte beim Verbinden mit dateibasierten ODBC- und OLE DB-Datenverbindungen

ODBC und OLE DB-Datenverbindungen mit dateibasierten Treibern legen den Pfad zur verbundenen Datendatei im Verbindungsstring frei. Der Pfad kann offen gelegt werden, wenn die Verbindung im Datenauswahldialog oder in bestimmten SQL-Abfragen bearbeitet wird. Dies gilt sowohl für den Standardmodus als auch den Legacymodus.

Wenn das Offenlegen des Pfads zur Datendatei nicht gewünscht wird, empfiehlt es sich, die Datendatei mit einer Ordner-Datenverbindung zu verbinden, falls möglich.

9.2 Einschränkungen im Standardmodus

Mehrere Befehle, Variablen und Funktionen können im Standardmodus nicht verwendet werden oder sind nur eingeschränkt nutzbar. Die Verwendung nicht unterstützter Befehle im Datenladeskript generiert einen Fehler beim Ausführen des Ladeskripts. Fehlermeldungen sind in der Skriptprotokolldatei enthalten. Die Verwendung nicht unterstützter Variablen und Funktionen erzeugt keine Nachrichten oder Protokolldateieinträge. Stattdessen liefert die Funktion NULL.

Es wird kein Hinweis darauf angezeigt, dass eine Variable, ein Befehl oder eine Funktion nicht unterstützt werden, wenn Sie das Datenladeskript bearbeiten.

Systemvariablen

Systemvariablen

Variable	Standardmodus	Legacymodus	Definition
Floppy	Nicht unterstützt	Unterstützt	Liefert die Laufwerksbezeichnung des ersten gefundenen Diskettenlaufwerks, in der Regel *a:*.
CD	Nicht unterstützt	Unterstützt	Liefert die Laufwerksbezeichnung des ersten gefundenen CD-ROM-Laufwerks. Wird kein CD-ROM-Laufwerk gefunden, wird *c:* ausgegeben.
QvPath	Nicht unterstützt	Unterstützt	Gibt den Pfad zur Qlik Sense-Programmdatei zurück.
QvRoot	Nicht unterstützt	Unterstützt	Gibt das Stammverzeichnis der Qlik Sense-Programmdatei zurück.
QvWorkPath	Nicht unterstützt	Unterstützt	Gibt den Pfad zur aktuellen Qlik Sense-App zurück.
QvWorkRoot	Nicht unterstützt	Unterstützt	Gibt das Stammverzeichnis der aktuellen Qlik Sense-App zurück.
WinPath	Nicht unterstützt	Unterstützt	Liefert den Pfad zu Windows.
WinRoot	Nicht unterstützt	Unterstützt	Liefert das Root-Verzeichnis von Windows.

Variable	Standardmodus	Legacymodus	Definition
$(include=...)	Unterstützte Eingabe: Pfad, der die Bibliotheksverbindung nutzt	Unterstützte Eingabe: Pfad, der die Bibliotheksverbindung oder das Dateisystem nutzt	Die **Include/Must_ Include**-Variable spezifiziert eine Datei, die in das Skript mit einbezogen und als Skript-Code evaluiert werden sollte. Sie wird nicht zum Hinzufügen von Daten verwendet. Sie können Teile Ihres Script-Codes in einer separaten Textdatei speichern und in verschiedenen Apps verwenden. Dies ist eine benutzerdefinierte Variable.

Reguläre Skriptbefehle

Reguläre Skriptbefehle

Befehl	Standardmodus	Legacymodus	Definition
Binary	Unterstützte Eingabe: Pfad, der die Bibliotheksverbindung nutzt	Unterstützte Eingabe: Pfad, der die Bibliotheksverbindung oder das Dateisystem nutzt	Der Befehl **binary** wird zum Laden von Daten aus einer anderen App verwendet.
Connect	Unterstützte Eingabe: Pfad, der die Bibliotheksverbindung nutzt	Unterstützte Eingabe: Pfad, der die Bibliotheksverbindung oder das Dateisystem nutzt	Der Befehl **CONNECT** legt den Qlik Sense-Zugriff auf eine allgemeine Datenbank über die OLE DB/ODBC-Schnittstelle fest. Für ODBC muss die Datenquelle zunächst mithilfe des ODBC-Administrators angegeben werden.

Befehl	Standardmodus	Legacymodus	Definition
Directory	Unterstützte Eingabe: Pfad, der die Bibliotheksverbindung nutzt	Unterstützte Eingabe: Pfad, der die Bibliotheksverbindung oder das Dateisystem nutzt	Die **Directory**-Anweisung definiert, welches Verzeichnis in den nachfolgenden **LOAD**-Anweisungen nach Datendateien durchsucht wird, bis eine neue **Directory**-Anweisung erstellt wird.
Execute	Nicht unterstützt	Unterstützte Eingabe: Pfad, der die Bibliotheksverbindung oder das Dateisystem nutzt	Der Befehl **Execute** wird zur Ausführung anderer Programme verwendet, während Qlik Sense Daten lädt. Dies dient z. B. dazu, notwendige Konvertierungen vorzunehmen.
LOAD from ...	Unterstützte Eingabe: Pfad, der die Bibliotheksverbindung nutzt	Unterstützte Eingabe: Pfad, der die Bibliotheksverbindung oder das Dateisystem nutzt	Der **LOAD**-Befehl lädt Felder aus einer Datei aus Daten, die im Skript definiert sind, aus einer zuvor geladenen Tabelle, aus einer Webseite, aus dem Ergebnis eines nachfolgenden **SELECT**-Befehls oder durch automatisches Generieren der Daten.
Store into ...	Unterstützte Eingabe: Pfad, der die Bibliotheksverbindung nutzt	Unterstützte Eingabe: Pfad, der die Bibliotheksverbindung oder das Dateisystem nutzt	Mit dem Befehl **Store** wird eine QVD-, Parquet-, CSV- oder TXT-Datei erstellt.

Steuerungsbefehle im Skript

Steuerungsbefehle im Skript

Befehl	Standardmodus	Legacymodus	Definition
For each... filelist mask/dirlist mask	Unterstützte Eingabe: Pfad, der die Bibliotheksverbindung nutzt Angezeigtes Ergebnis: Bibliotheksverbindung	Unterstützte Eingabe: Pfad, der die Bibliotheksverbindung oder das Dateisystem nutzt Angezeigtes Ergebnis: Bibliotheksverbindung oder Dateisystempfad, je nach Eingabe	Durch die Syntax filelist mask wird eine kommagetrennte Liste aller Dateien im aktuellen Verzeichnis generiert, die der **filelist mask**entsprechen. Durch die Syntax **dirlist mask** wird eine kommagetrennte Liste aller Verzeichnisse im aktuellen Verzeichnis generiert, die der Verzeichnisnamenmaske entsprechen.

Dateifunktionen

Dateifunktionen

Funktion	Standardmodus	Legacymodus	Definition
Attribute()	Unterstützte Eingabe: Pfad, der die Bibliotheksverbindung nutzt	Unterstützte Eingabe: Pfad, der die Bibliotheksverbindung oder das Dateisystem nutzt	Liefert den Wert der Metatags unterschiedlicher Mediendateiformate als Text.
ConnectString()	Angezeigtes Ergebnis: Bibliotheksverbindungsname	Bibliotheksverbindungsname oder tatsächliche Verbindung, je nach Eingabe	Liefert den aktiven connect-String für ODBC- oder OLE DB-Verbindungen.
FileDir()	Angezeigtes Ergebnis: Bibliotheksverbindung	Angezeigtes Ergebnis: Bibliotheksverbindung oder Dateisystempfad, je nach Eingabe	Die Funktion **FileDir** liefert den Pfad zum Verzeichnis der gerade eingelesenen Tabellendatei.

Funktion	Standardmodus	Legacymodus	Definition
FilePath()	Angezeigtes Ergebnis: Bibliotheksverbindung	Angezeigtes Ergebnis: Bibliotheksverbindung oder Dateisystempfad, je nach Eingabe	Die Funktion **FilePath** liefert den vollständigen Pfad zur gerade eingelesenen Tabellendatei.
FileSize()	Unterstützte Eingabe: Pfad, der die Bibliotheksverbindung nutzt	Unterstützte Eingabe: Pfad, der die Bibliotheksverbindung oder das Dateisystem nutzt	Die Funktion **FileSize** liefert eine ganze Zahl, die die Größe der Datei filename in Byte angibt. Ist filename nicht angegeben, wird die Größe der gerade eingelesenen Tabellendatei ausgegeben.
FileTime()	Unterstützte Eingabe: Pfad, der die Bibliotheksverbindung nutzt	Unterstützte Eingabe: Pfad, der die Bibliotheksverbindung oder das Dateisystem nutzt	Die Funktion **FileTime** gibt einen Zeitstempel im UTC-Format für die letzte Änderung einer angegebenen Datei zurück. Wenn keine Datei angegeben wird, gibt die Funktion einen Zeitstempel im UTC-Format für die letzte Änderung an der aktuell gelesenen Tabellendatei zurück.

Funktion	Standardmodus	Legacymodus	Definition
GetFolderPath()	Nicht unterstützt	Angezeigtes Ergebnis: Absoluter Pfad	Die Funktion **GetFolderPath** liefert den Wert der Microsoft Windows *SHGetFolderPath*-Funktion. Diese Funktion nimmt als Eingabe den Namen eines Microsoft Windows -Ordners und liefert den vollständigen Pfad des Ordners.
QvdCreateTime()	Unterstützte Eingabe: Pfad, der die Bibliotheksverbindung nutzt	Unterstützte Eingabe: Pfad, der die Bibliotheksverbindung oder das Dateisystem nutzt	Diese Skriptfunktion gibt den im XML-Header der QVD-Datei gespeicherten Zeitstempel zurück, sofern dieser in der Datei vorhanden ist, ansonsten gibt sie sie das Ergebnis NULL zurück. Im Zeitstempel wird die Uhrzeit im UTC-Format angegeben.
QvdFieldName()	Unterstützte Eingabe: Pfad, der die Bibliotheksverbindung nutzt	Unterstützte Eingabe: Pfad, der die Bibliotheksverbindung oder das Dateisystem nutzt	Diese Skriptfunktion liefert den Namen von Feld Nummer **fieldno** in einer QVD-Datei. Ist das Feld nicht vorhanden, liefert diese Funktion NULL.
QvdNoOfFields()	Unterstützte Eingabe: Pfad, der die Bibliotheksverbindung nutzt	Unterstützte Eingabe: Pfad, der die Bibliotheksverbindung oder das Dateisystem nutzt	Diese Skriptfunktion liefert die Zahl der Felder in einer QVD-Datei.
QvdNoOfRecords()	Unterstützte Eingabe: Pfad, der die Bibliotheksverbindung nutzt	Unterstützte Eingabe: Pfad, der die Bibliotheksverbindung oder das Dateisystem nutzt	Diese Skriptfunktion liefert die Zahl der Datensätze in einer QVD-Datei.

Funktion	Standardmodus	Legacymodus	Definition
QvdTableName()	Unterstützte Eingabe: Pfad, der die Bibliotheksverbindung nutzt	Unterstützte Eingabe: Pfad, der die Bibliotheksverbindung oder das Dateisystem nutzt	Diese Skriptfunktion liefert den Namen der in der QVD-Datei gespeicherten Tabelle.

Systemfunktionen

Systemfunktionen

Funktion	Standardmodus	Legacymodus	Definition
DocumentPath()	Nicht unterstützt	Angezeigtes Ergebnis: Absoluter Pfad	Diese Funktion gibt einen String mit dem vollständigen Pfad zur aktuellen Qlik Sense-App zurück.
GetRegistryString()	Nicht unterstützt	Unterstützt	Liefert den Wert des angegebenen Registry-Keys unter dem angegebenen Registry-Pfad. Diese Funktion kann im Skript und im Diagramm verwendet werden.

9.3 Deaktivieren des Standardmodus

Sie können den Standardmodus deaktivieren (gleichbedeutend mit der Aktivierung des Legacymodus), um QlikView-Ladeskripte wiederverwenden zu können, die sich auf absolute oder relative Dateipfade sowie Bibliotheksverbindungen beziehen.

Die Deaktivierung des Standardmodus kann ein Sicherheitsrisiko darstellen, da das Dateisystem offengelegt wird.

Qlik Sense

Bei Qlik Sense lässt sich der Standardmodus in QMC über die Eigenschaft **Standardmodus** deaktivieren.

Qlik Sense Desktop

In Qlik Sense Desktop können Sie den Standard-/Legacymodus in *Settings.ini* einstellen.

Wenn Sie Qlik Sense Desktop im Standardinstallationsverzeichnis installiert haben, befindet sich *Settings.ini* unter *C:\Users\{user}\Documents\Qlik\Sense\Settings.ini*. Wenn Sie Qlik Sense Desktop in einem selbst ausgewählten Ordner installiert haben, befindet sich *Settings.ini* im *Engine*-Ordner des Installationspfads.

Gehen Sie folgendermaßen vor:

1. Öffnen Sie *Settings.ini* in einem Texteditor.
2. Ändern Sie *StandardReload=1* zu *StandardReload=0*.
3. Speichern Sie die Datei und starten Sie Qlik Sense Desktop.

Qlik Sense Desktop wird nun im Legacymodus ausgeführt.

Einstellungen

Die verfügbaren Einstellungen für StandardReload sind:

- 1 (Standardmodus)
- 0 (Legacymodus)

10 Skripterstellung auf Diagrammebene

Wenn Sie Diagrammdaten ändern, verwenden Sie einen Teil des Qlik Sense-Skripts, der aus einer Reihe von Befehlen besteht. Ein Befehl kann ein regulärer Skriptbefehl oder Steuerungsbefehl sein. Manchen Befehlen kann ein Zusatz voran- oder nachgestellt werden.

Gewöhnliche Befehle dienen dazu, Daten einzulesen und diese Daten zu strukturieren oder zu verändern. Sie können sich über mehrere Zeilen erstrecken und müssen stets mit einem Semikolon enden , ";".

Steuerungsbefehle steuern die Ausführung des Skripts. Sie dürfen nicht über eine Zeile hinausgehen und werden durch ein Semikolon oder eine Zeilenschaltung beendet.

Eine Ergänzung durch Zusätze ist nur für gewöhnliche Befehle, nicht aber für Steuerungsbefehle möglich.

Sämtliche Skriptbefehle können in Groß- oder Kleinbuchstaben oder einer Kombination aus beiden eingegeben werden. Bei Feld- und Variablennamen, die im Skript vorkommen, wird Groß- und Kleinschreibung jedoch unterschieden.

Dieser Abschnitt enthält eine alphabetische Liste aller Skriptanweisungen, Steuerungsanweisungen und Zusätze, die im Teil des Skripts bei der Änderung von Diagrammdaten verwendet werden.

10.1 Steuerungsbefehle

Wenn Sie Diagrammdaten ändern, verwenden Sie einen Teil des Qlik Sense-Skripts, der aus einer Reihe von Befehlen besteht. Ein Befehl kann ein regulärer Skriptbefehl oder Steuerungsbefehl sein.

Steuerungsbefehle steuern die Ausführung des Skripts. Sie dürfen nicht über eine Zeile hinausgehen und werden durch ein Semikolon oder eine Zeilenschaltung beendet.

Zusätze werden nie auf Steuerungsbefehle angewandt.

Sämtliche Skriptbefehle können in Groß- oder Kleinbuchstaben oder einer Kombination aus beiden eingegeben werden.

Steuerungsbefehle zur Diagrammänderung – Übersicht

Jede Funktion wird nach der Übersicht genauer beschrieben. Sie können auch auf den Funktionsnamen in der Syntax klicken, um direkt auf die Details zu der spezifischen Funktion zuzugreifen.

Call

Der Steuerungsbefehl **call** ruft eine Subroutine auf, die vorher durch einen **sub**-Befehl definiert werden muss.

```
Call name ( [ paramlist ])
```

Do..loop

Der Steuerungsbefehl **do..loop** definiert eine Skriptiteration, die einen bzw. mehrere Befehle ausführt, bis eine logische Bedingung erfüllt ist.

```
Do..loop [ ( while | until ) condition ] [statements]
[exit do [ ( when | unless ) condition ] [statements]
loop [ ( while | until ) condition ]
```

End

Das Skriptschlüsselwort **End** wird zum Abschließen der Bedingungen **If**, **Sub** und **Switch** verwendet.

Exit

Das Skriptschlüsselwort **Exit** ist Teil des Befehls **Exit Script**, kann aber auch zum Beenden der Bedingungen **Do**, **For** oder **Sub** verwendet werden.

Exit script

Dieser Steuerungsbefehl beendet die Ausführung des Skripts. Er kann an beliebiger Stelle des Skripts stehen.

```
Exit script[ (when | unless) condition ]
```

For..next

Der Steuerungsbefehl **for..next** ist eine Skriptiteration mit einem Zähler. Für jeden Zähler innerhalb der festgelegten Grenzen werden die Befehle innerhalb der Schleife, die durch **for**und **next** eingeschlossen sind, jeweils einmal ausgeführt.

```
For..next  counter = expr1 to expr2 [ stepexpr3 ]

[statements]

[exit for [ ( when | unless ) condition ]

[statements]

Next [counter]
```

For each ..next

Der Steuerungsbefehl **for each..next** definiert eine Skriptiteration, die für jeden Wert in einer kommagetrennten Liste einen oder mehrere Befehle ausführt. Für jeden Wert der Liste werden die Befehle zwischen **for** und **next** einmal ausgeführt.

```
For each..next var in list

[statements]

[exit for [ ( when | unless ) condition ]

[statements]

next [var]
```

If..then

Der Steuerungsbefehl **if..then** ist eine Skriptauswahl, mit der die Ausführung des Skripts gezwungen wird, abhängig von einer oder mehreren logischen Bedingungen unterschiedlichen Pfaden zu folgen.

*Da **if..then** zu den Steuerungsbefehlen gehört und mit einem Semikolon oder einer Zeilenschaltung abschließt, dürfen sich die einzelnen Befehlssequenzen (**if..then**, **elseif..then**, **else** und **end if**) nicht über mehrere Zeilen erstrecken.*

```
If..then..elseif..else..end if condition then

  [ statements ]

{ elseif condition then

  [ statements ] }

[ else

  [ statements ] ]

end if
```

Next
Das Skriptschlüsselwort **Next** wird zum Schließen von zirkulären **For**-Verknüpfungen verwendet.

Sub
Der Steuerungsbefehl **sub..end sub** definiert eine Subroutine, die zu einem späteren Zeitpunkt durch den Befehl **call** aufgerufen werden kann.

```
Sub..end sub name [ ( paramlist )] statements end sub
```

Switch
Der Steuerungsbefehl **switch** ist eine Skriptauswahl, mit der die Ausführung des Skripts gezwungen wird, abhängig vom Wert einer Formel unterschiedlichen Pfaden zu folgen.

```
Switch..case..default..end switch expression {case valuelist [ statements ]}
[default statements] end switch
```

To
Das Skriptschlüsselwort **To** wird in verschiedenen Skriptbefehlen verwendet.

Call

Der Steuerungsbefehl **call** ruft eine Subroutine auf, die vorher durch einen **sub**-Befehl definiert werden muss.

Syntax:

```
Call name ( [ paramlist ])
```

Argumente:

Argumente

Argument	Beschreibung
name	Der Name der Subroutine.
paramlist	Eine durch Komma getrennte Liste der Parameter, die an die Subroutine übergeben werden. Jeder Eintrag dieser Liste kann ein Feldname, eine Variable oder eine beliebige Formel sein.

Die Subroutine, die von einem **call**-Befehl aus aufgerufen wird, muss bereits an einer vorherigen Stelle im Skript durch ein **sub** definiert sein.

Die Parameter werden dabei in die Subroutine kopiert und, sofern es sich beim Parameter im **call**-Befehl um eine Variable und nicht um eine Formel handelt, beim Beenden der Subroutine auch wieder zurückkopiert.

Beschränkungen:

- Da **call** zu den Steuerungsbefehlen gehört und daher mit einem Semikolon oder einer Zeilenschaltung abschließt, darf sich der Befehl nicht über mehrere Zeilen erstrecken.
- Wenn Sie eine Unterroutine mit `sub..end sub` innerhalb eines Steuerungsbefehls definieren, beispielsweise `if..then`, können Sie nur die Unterroutine innerhalb des gleichen Steuerungsbefehls aufrufen.

Do..loop

Der Steuerungsbefehl **do..loop** definiert eine Skriptiteration, die einen bzw. mehrere Befehle ausführt, bis eine logische Bedingung erfüllt ist.

Syntax:

```
Do [ ( while | until ) condition ] [statements]
[exit do [ ( when | unless ) condition ] [statements]
loop[ ( while | until ) condition ]
```

*Da **do..loop** zu den Steuerungsbefehlen gehört und mit einem Semikolon oder einer Zeilenschaltung abschließt, dürfen sich die einzelnen Befehlssequenzen (**do**, **exit do** und **loop**) nicht über mehrere Zeilen erstrecken.*

Argumente:

Argumente

Argument	Beschreibung
condition	Eine logische Formel, die True oder False ergibt.
statements	Jede Gruppe von einem oder mehreren Qlik Sense-Skriptbefehlen.
while / until	Die **while**- oder **until**-Bedingungen dürfen nur einmal im **do..loop**-Befehl auftreten, d. h. entweder nach dem **do** oder nach dem **loop**. Die logische Prüfung der Bedingung findet für jeden Durchlauf der Schleife jeweils nur einmal statt, auch wenn die Bedingung nochmals in der Schleife erscheint.
exit do	Steht in der Schleife eine **exit do**-Bedingung, wird die Ausführung des Skripts beim ersten Befehl nach der Schleife, d. h. nach der Zeile mit dem abschließenden **loop**, fortgesetzt. Auf **exit do** kann verzichtet werden, wenn stattdessen **when** oder **unless** verwendet wird.

End

Das Skriptschlüsselwort **End** wird zum Abschließen der Bedingungen **If**, **Sub** und **Switch** verwendet.

Exit

Das Skriptschlüsselwort **Exit** ist Teil des Befehls **Exit Script**, kann aber auch zum Beenden der Bedingungen **Do**, **For** oder **Sub** verwendet werden.

Exit script

Dieser Steuerungsbefehl beendet die Ausführung des Skripts. Er kann an beliebiger Stelle des Skripts stehen.

Syntax:

```
Exit Script [ (when | unless) condition ]
```

Da **exit script** zu den Steuerungsbefehlen gehört und daher mit einem Semikolon oder einer Zeilenschaltung abschließt, darf sich der Befehl nicht über mehrere Zeilen erstrecken.

Argumente:

Argumente

Argument	Beschreibung
condition	Eine logische Formel, die True oder False ergibt.
when / unless	Auf den **exit script**-Zusatz kann verzichtet werden, wenn stattdessen **when** oder **unless** verwendet wird.

Beispiele:

```
//Exit script
Exit Script;

//Exit script when a condition is fulfilled
Exit Script when a=1
```

For..next

Der Steuerungsbefehl **for..next** ist eine Skriptiteration mit einem Zähler. Für jeden Zähler innerhalb der festgelegten Grenzen werden die Befehle innerhalb der Schleife, die durch **for**und **next** eingeschlossen sind, jeweils einmal ausgeführt.

Syntax:

```
For counter = expr1 to expr2 [ step expr3 ]

[statements]

[exit for [ ( when | unless ) condition ]

[statements]

Next [counter]
```

Die Formeln *expr1*, *expr2* und *expr3* werden nur beim ersten Durchlauf der Schleife ausgewertet. Der Wert der Zählervariablen kann durch Befehle innerhalb der Schleife geändert werden, dies ist aber nicht empfehlenswert.

Steht in der Schleife eine **exit for**-Bedingung, wird die Ausführung des Skripts beim ersten Befehl nach der Schleife, d. h. nach der Zeile mit dem abschließenden **next**, fortgesetzt. Auf **exit for** kann verzichtet werden, wenn stattdessen **when** oder **unless** verwendet wird.

> *Da **for..next** zu den Steuerungsbefehlen gehört und mit einem Semikolon oder einer Zeilenschaltung abschließt, dürfen sich die einzelnen Befehlssequenzen (**for..to..step**, **exit for** und **next**) nicht über mehrere Zeilen erstrecken.*

Argumente:

Argumente

Argument	Beschreibung
counter	Ein Variablenname. Ist *counter* nach **next** festgelegt, muss sie denselben Variablennamen haben, wie den hinter dem zugehörigen **for**.

Argument	Beschreibung
expr1	Eine Formel, deren Ergebnis den ersten Wert der Variable *counter* ergibt, für welche die Schleife ausgeführt wird.
expr2	Eine Formel, deren Ergebnis den letzten Wert der Variable *counter* ergibt, für welche die Schleife ausgeführt wird.
expr3	Eine Formel, deren Ergebnis den Zuwachs von *counter* ergibt, um den sich der Zähler bei jedem Durchlauf der Schleife erhöht.
condition	Eine logische Formel, die True oder False ergibt.
statements	Jede Gruppe von einem oder mehreren Qlik Sense-Skriptbefehlen.

For each..next

Der Steuerungsbefehl **for each..next** definiert eine Skriptiteration, die für jeden Wert in einer kommagetrennten Liste einen oder mehrere Befehle ausführt. Für jeden Wert der Liste werden die Befehle zwischen **for** und **next** einmal ausgeführt.

Syntax:
Durch eine besondere Syntax ist es möglich, eine Liste mit Verzeichnis- und Dateinamen zu generieren.

```
for each var in list

[statements]

[exit for [ ( when | unless ) condition ]

[statements]

next [var]
```

Argumente:

Argumente

Argument	Beschreibung
var	Eine Skriptvariable, die bei jedem Durchlauf der Schleife den jeweils nächsten Wert in der Werteliste annimmt. Ist **var** nach **next** festgelegt, muss sie denselben Variablennamen haben, wie den hinter dem zugehörigen **for each**.

Der Wert von **var** kann durch Befehle innerhalb der Schleife geändert werden, dies ist aber nicht empfehlenswert.

Steht in der Schleife eine **exit for**-Bedingung, wird die Ausführung des Skripts beim ersten Befehl nach der Schleife, d. h. nach der Zeile mit dem abschließenden **next**, fortgesetzt. Auf **exit for** kann verzichtet werden, wenn stattdessen **when** oder **unless** verwendet wird.

*Da **for each..next** zu den Steuerungsbefehlen gehört und mit einem Semikolon oder einer Zeilenschaltung abschließt, dürfen sich die einzelnen Befehlssequenzen (**for each**, **exit for** und **next**) nicht über mehrere Zeilen erstrecken.*

Syntax:

```
list := item { , item }
```

```
item := constant | (expression) | filelist mask | dirlist mask |
fieldvaluelist mask
```

Argumente

Argument	Beschreibung
constant	Beliebige Zahl oder String. Beachten Sie, dass ein direkt im Skript geschriebener String zwischen einfachen Anführungszeichen stehen muss. Ein String ohne einfache Anführungszeichen wird als Variable interpretiert. Anschließend wird der Wert der Variable verwendet. Zahlen müssen nicht zwischen einfachen Anführungszeichen stehen.
expression	Eine beliebige Formel.
mask	Eine Vorgabe für Dateinamen bzw. Ordnernamen, die alle in Dateinamen zugelassenen Zeichen enthalten kann, sowie die Wildcards ***** und **?**. Sie können absolute Dateipfade oder lib://-Pfade verwenden.
condition	Eine logische Formel, die True oder False ergibt.
statements	Jede Gruppe von einem oder mehreren Qlik Sense-Skriptbefehlen.
filelist mask	Durch die Syntax wird eine kommagetrennte Liste aller Dateien im aktuellen Verzeichnis generiert, die der Dateinamenmaske entsprechen. *Dieses Argument unterstützt im Standardmodus nur Bibliotheksverbindungen.*
dirlist mask	Durch die Syntax wird eine kommagetrennte Liste aller Ordner im aktuellen Ordner generiert, die der Ordnernamenmaske entsprechen. *Dieses Argument unterstützt im Standardmodus nur Bibliotheksverbindungen.*
fieldvaluelist mask	Diese Syntax durchläuft die Werte eines Felds, das bereits in Qlik Sense geladen wurde.

*Qlik Konnektoren für Webspeicher-Anbieter und andere DataFiles-Verbindungen unterstützen keine Filtermasken, die Platzhalterzeichen (***** und **?**) enthalten.*

Example 1: Laden einer Liste von Dateien

```
// LOAD the files 1.csv, 3.csv, 7.csv and xyz.csv
for each a in 1,3,7,'xyz'
   LOAD * from file$(a).csv;
next
```

Example 2: Laden einer Dateiliste auf Speichermedium

In diesem Beispiel wird eine Liste aller relevanten Qlik Sense-Dateien in einen Ordner geladen.

```
sub DoDir (Root)
   for each Ext in 'qvw', 'qva', 'qvo', 'qvs', 'qvc', 'qvf', 'qvd'

      for each File in filelist (Root&'/*.' &Ext)

         LOAD
            '$(File)' as Name,
            FileSize( '$(File)' ) as Size,
            FileTime( '$(File)' ) as FileTime
         autogenerate 1;

      next File

   next Ext
   for each Dir in dirlist (Root&'/*' )

      call DoDir (Dir)

   next Dir

end sub

call DoDir ('lib://DataFiles')
```

Example 3: Aggregierung nach den Werten eines Felds

In diesem Beispiel wird die Liste der geladenen Werte von FIELD aggregiert und ein neues Feld – NEWFIELD – erstellt. Für jeden Wert von FIELD werden zwei Datensätze NEWFIELD erstellt.

```
load * inline [
FIELD
one
two
three
];

FOR Each a in FieldValueList('FIELD')
LOAD '$(a)' &'-'&RecNo() as NEWFIELD AutoGenerate 2;
NEXT a
```

Die sich ergebende Tabelle sieht folgendermaßen aus:

Example table

NEWFIELD
one-1
one-2
two-1
two-2
three-1
three-2

If..then..elseif..else..end if

Der Steuerungsbefehl **if..then** ist eine Skriptauswahl, mit der die Ausführung des Skripts gezwungen wird, abhängig von einer oder mehreren logischen Bedingungen unterschiedlichen Pfaden zu folgen.

Steuerungsbefehle steuern den Ablauf der Skriptausführung. In einer Diagrammformel verwenden Sie stattdessen die Konditionalfunktion **if**.

Syntax:

```
If  condition then

  [ statements ]

{ elseif condition then

  [ statements ] }

[ else

  [ statements ] ]

end if
```

Da **if..then** zu den Steuerungsbefehlen gehört und mit einem Semikolon oder einer Zeilenschaltung abschließt, dürfen sich die einzelnen Befehlssequenzen (**if..then**, **elseif..then**, **else** und **end if**) nicht über mehrere Zeilen erstrecken.

Argumente:

Argumente

Argument	Beschreibung
condition	Eine logische Formel, die True oder False ergibt.
statements	Jede Gruppe von einem oder mehreren Qlik Sense-Skriptbefehlen.

Example 1:

```
if a=1 then
      LOAD * from abc.csv;

      SQL SELECT e, f, g from tab1;
end if
```

Example 2:

```
if a=1 then; drop table xyz; end if;
```

Example 3:

```
if x>0 then
      LOAD * from pos.csv;
elseif x<0 then
      LOAD * from neg.csv;
else
      LOAD * from zero.txt;
end if
```

Next

Das Skriptschlüsselwort **Next** wird zum Schließen von zirkulären **For**-Verknüpfungen verwendet.

Sub..end sub

Der Steuerungsbefehl **sub..end sub** definiert eine Subroutine, die zu einem späteren Zeitpunkt durch den Befehl **call** aufgerufen werden kann.

Syntax:

```
Sub name [ ( paramlist )] statements end sub
```

Argumente werden in die Subroutine kopiert. Wenn es sich bei dem entsprechend aufgeführten Parameter im **call**-Befehl um einen Variablennamen handelt, werden sie beim Beenden der Subroutine wieder zurückkopiert.

Wenn in einer Subroutine mehr Parameter definiert sind, als aus dem **call**-Befehl übernommen werden, erhalten die übrigen Parameter den NULL-Wert und können als lokale Variable in der Subroutine verwendet werden.

Argumente:

Argumente

Argument	Beschreibung
name	Der Name der Subroutine.
paramlist	Eine kommagetrennte Liste von Variablen, die als Parameter in der Subroutine dienen. Sie können wie jede Variable innerhalb der Subroutine verwendet werden.
statements	Jede Gruppe von einem oder mehreren Qlik Sense-Skriptbefehlen.

Beschränkungen:

- Da **sub** zu den Steuerungsbefehlen gehört und mit einem Semikolon oder einer Zeilenschaltung abschließt, dürfen sich die einzelnen Befehlssequenzen (**sub** und **end sub**) nicht über mehrere Zeilen erstrecken.
- Wenn Sie eine Unterroutine mit `Sub..end sub` innerhalb eines Steuerungsbefehls definieren, beispielsweise `if..then`, können Sie nur die Unterroutine innerhalb des gleichen Steuerungsbefehls aufrufen.

Example 1:

```
Sub INCR (I,J)

I = I + 1

Exit Sub when I < 10

J = J + 1

End Sub

Call INCR (X,Y)
```

Example 2: - Übernahme von Parametern

```
Sub ParTrans (A,B,C)

A=A+1

B=B+1

C=C+1

End Sub

A=1

X=1
```

```
C=1

Call ParTrans (A, (X+1)*2)
```

Das Ergebnis des obigen Vorgangs besteht darin, dass A lokal, innerhalb der Subroutine, den Wert 1 erhält, B den Wert 4 und C den Wert NULL.

Beim Abschluss der Subroutine erhält die globale Variable A den Wert 2 (aus der Subroutine zurückkopiert). Der zweite aufgeführte Parameter „(X+1)*2“ wird nicht zurückkopiert, da es sich nicht um eine Variable handelt. Die globale Variable C wird vom Subroutinen-Call nicht beeinflusst.

Switch..case..default..end switch

Der Steuerungsbefehl **switch** ist eine Skriptauswahl, mit der die Ausführung des Skripts gezwungen wird, abhängig vom Wert einer Formel unterschiedlichen Pfaden zu folgen.

Syntax:

```
Switch expression {case valuelist [ statements ]} [default statements] end switch
```

> *Da **switch** zu den Steuerungsbefehlen gehört und mit einem Semikolon oder einer Zeilenschaltung abschließt, dürfen sich die einzelnen Befehlssequenzen (**switch**, **case**, **default** und **end switch**) nicht über mehrere Zeilen erstrecken.*

Argumente:

Argumente

Argument	Beschreibung
expression	Eine beliebige Formel.
valuelist	Eine durch Komma getrennte Liste von Werten, die mit dem Ergebnis der Formel verglichen wird. Die Ausführung des Skripts wird bei den Befehlen fortgesetzt, in deren zugehöriger Werteliste als Erstes eine Übereinstimmung mit dem Ergebnis der Formel festgestellt wird. Die Werte in der Werteliste können beliebige Formeln sein. Wird in keiner der **case**-Bedingungen eine Übereinstimmung festgestellt, werden die auf **default** folgenden Befehle ausgeführt, sofern vorhanden.
statements	Jede Gruppe von einem oder mehreren Qlik Sense-Skriptbefehlen.

Beispiel:

```
Switch I

Case 1

LOAD '$(I): CASE 1' as case autogenerate 1;

Case 2

LOAD '$(I): CASE 2' as case autogenerate 1;
```

```
Default

LOAD '$(I): DEFAULT' as case autogenerate 1;

End Switch
```

To

Das Skriptschlüsselwort **To** wird in verschiedenen Skriptbefehlen verwendet.

10.2 Zusätze

Eine Ergänzung durch Zusätze ist nur für gewöhnliche Befehle, nicht aber für Steuerungsbefehle möglich.

Sämtliche Skriptbefehle können in Groß- oder Kleinbuchstaben oder einer Kombination aus beiden eingegeben werden. Bei Feld- und Variablennamen, die im Skript vorkommen, wird Groß- und Kleinschreibung jedoch unterschieden.

Zusätze zur Diagrammänderung – Übersicht

Jede Funktion wird nach der Übersicht genauer beschrieben. Sie können auch auf den Funktionsnamen in der Syntax klicken, um direkt auf die Details zu der spezifischen Funktion zuzugreifen.

Add
Der Zusatz **Add** kann zu jedem **LOAD**- oder **SELECT**-Befehl im Skript hinzugefügt werden, um anzugeben, dass Datensätze zu einer anderen Tabelle hinzugefügt werden sollen. Er gibt auch an, dass dieser Befehl in einem partiellen Ladevorgang ausgeführt werden soll. Der Zusatz **Add** kann auch in einem **Map**-Befehl verwendet werden.

```
Add [only] [Concatenate[(tablename )]] (loadstatement | selectstatement)
Add [ Only ] mapstatement
```

Replace
Der Zusatz **Replace** kann zu jedem **LOAD**- oder **SELECT**-Befehl im Skript hinzugefügt werden, um anzugeben, dass die geladene Tabelle eine andere Tabelle ersetzen soll. Er gibt auch an, dass dieser Befehl in einem partiellen Ladevorgang ausgeführt werden soll. Der Zusatz **Replace** kann auch in einem **Map**-Befehl verwendet werden.

```
Replace [only] [Concatenate[(tablename) ]] (loadstatement | selectstatement)
Replace [only] mapstatement
```

Add

In einem Diagrammänderungskontext wird der Zusatz **Add** mit **LOAD** verwendet, um Werte an die Tabelle *HC1* anzuhängen und den von der Qlik Associative Engine berechneten Hypercube darzustellen. Sie können eine oder mehrere Spalten angeben. Fehlende Werte werden automatisch von der Qlik Associative Engine ausgefüllt.

Syntax:

```
Add loadstatement
```

Beispiel:

In diesem Beispiel werden zwei Zeilen zu den Spalten *Dates* und *Sales* über den Inline-Befehl hinzugefügt

```
Add Load
x as Dates,
y as Sales
Inline
[
Dates,Sales
2001/09/1,1000
2001/09/10,-300
]
```

Replace

In einem Diagrammänderungskontext ändert der Zusatz **Replace** alle Werte für die Tabelle *HC1* mit einem vom Skript definierten berechneten Wert.

Syntax:

```
Replace loadstatement
```

Beispiel:

Mit diesem Beispiel werden alle Werte in der Spalte `z` mit der Summe von `x` und `y` überschrieben.

```
Replace Load
x+y as z
Resident HC1;
```

10.3 Reguläre Anweisungen

Gewöhnliche Befehle dienen dazu, Daten einzulesen und diese Daten zu strukturieren oder zu verändern. Sie können sich über mehrere Zeilen erstrecken und müssen stets mit einem Semikolon enden , ";".

Sämtliche Skriptbefehle können in Groß- oder Kleinbuchstaben oder einer Kombination aus beiden eingegeben werden. Bei Feld- und Variablennamen, die im Skript vorkommen, wird Groß- und Kleinschreibung jedoch unterschieden.

Reguläre Anweisungen zur Diagrammänderung – Übersicht

Jede Funktion wird nach der Übersicht genauer beschrieben. Sie können auch auf den Funktionsnamen in der Syntax klicken, um direkt auf die Details zu der spezifischen Funktion zuzugreifen.

LOAD

In einem Diagrammänderungskontext lädt die **LOAD**-Anweisung zusätzliche Daten aus im Skript definierten Daten oder aus einer zuvor geladenen Tabelle in den Hypercube. Daten können auch aus Analyseverbindungen geladen werden.

*Die **LOAD**-Anweisung muss den Zusatz **Replace** oder **Add** haben, andernfalls wird sie abgelehnt.*

```
Add | Replace Load [ distinct ] fieldlist

(

inline data [ format-spec ] |

resident table-label

) | extension pluginname.functionname([script] tabledescription)]

[ where criterion | while criterion ]

[ group by groupbyfieldlist ]

[order by orderbyfieldlist ]
```

Let

Der **let**-Befehl ergänzt den **set**-Befehl und definiert die Skriptvariablen. Im Gegensatz zum **set**-Befehl wird beim **let**-Befehl der Ausdruck rechts des Gleichheitszeichens "=" zur Laufzeit des Skripts ausgewertet, bevor er der Variablen zugewiesen wird.

```
Let variablename=expression
```

Set

Der Befehl **set** wird zum Festlegen der Skriptvariablen verwendet. Diese können Strings, Pfade, Laufwerke usw. im Skript ersetzen.

```
Set variablename=string
```

Put

Die **Put**-Anweisung wird verwendet, um einen numerischen Wert im Hypercube festzulegen.

HCValue

Die **HCValue**-Anweisung wird verwendet, um Werte in einer Zeile einer angegebenen Spalte abzurufen.

Load

In einem Diagrammänderungskontext lädt die **LOAD**-Anweisung zusätzliche Daten aus im Skript definierten Daten oder aus einer zuvor geladenen Tabelle in den Hypercube. Daten können auch aus Analyseverbindungen geladen werden.

*Die **LOAD**-Anweisung muss den Zusatz **Replace** oder **Add** haben, andernfalls wird sie abgelehnt.*

Syntax:

```
Add | Replace LOAD fieldlist
```

```
(

inline data [ format-spec ] |

resident table-label

) | extension pluginname.functionname([script] tabledescription)]

[ where criterion | while criterion ]

[ group by groupbyfieldlist ]

[order by orderbyfieldlist ]
```

Argumente:

Argumente

Argument	Beschreibung
fieldlist	*fieldlist ::= (* | field{**,** * | field })* Liste der zu ladenden Felder. * in einer Felderliste bedeutet alle Felder in der Tabelle. *field ::= (fieldref | expression) [***as** *aliasname]* Die Felddefinition muss immer eine Literale enthalten, einen Verweis auf ein bestehendes Feld oder eine Formel. *fieldref ::= (fieldname |@fieldnumber |@startpos:endpos [* ***I*** | ***U*** | ***R*** | ***B*** | **T**]) Dabei ist *fieldname* ein Text, der einem Feldnamen in der Tabelle entspricht. Beachten Sie, dass der Feldname zwischen geraden doppelten Anführungszeichen oder eckigen Klammern stehen muss, wenn er beispielsweise Leerzeichen enthält. Mitunter sind Feldnamen nicht im Tabellenkopf verfügbar. Verwenden Sie in diesem Fall stattdessen folgende Notation: *@fieldnumber* bezeichnet die Nummer des Feldes in einer Textdatei mit Trennzeichen. Es muss eine ganze positive Zahl mit vorangehendem "@" sein. Die Nummerierung beginnt stets mit 1 und endet mit der Gesamtzahl der Felder. *@startpos*:*endpos* bezeichnet Start- und Endposition eines Feldes in einer Datei mit festen Satzlängen. Die Positionen müssen beide positive ganze Zahlen sein. Beiden Zahlen muss ein "@" vorangehen und sie müssen durch einen Doppelpunkt getrennt sein. Die Nummerierung beginnt stets mit 1 und geht bis zu der entsprechenden Zahl von Stellen. Im letzten Feld wird **n** als Endposition verwendet. • Folgt direkt hinter *@startpos*:*endpos* der Buchstabe **I** oder **U**, werden die eingelesenen Daten als binär codierte ganze Zahl mit Vorzeichen (**I**) bzw. als binär codierte ganze Zahl ohne Vorzeichen **U** (Intel Byte Order) interpretiert. Die Zahl der eingelesenen Positionen muss 1, 2 oder 4 betragen. • Folgt direkt hinter *@startpos*:*endpos* der Buchstabe **R**, werden die eingelesenen Daten als binär codierte reelle Zahlen (IEEE 32-Bit- oder 64-Bit-Gleitkommazahl) interpretiert. Die Zahl der eingelesenen Positionen muss 4 oder 8 betragen. • Folgt direkt hinter *@startpos*:*endpos* der Buchstabe **B**, werden die eingelesenen Daten als binär codierte Dezimalzahlen BCD (Binary Coded Decimal) entsprechend dem COMP-3-Standard interpretiert. Es kann eine beliebige Zahl von Bytes angegeben werden. *expression* kann eine numerische Funktion oder eine Stringfunktion sein, die sich auf ein oder mehrere Felder derselben Tabelle bezieht. Weitere Informationen finden Sie in den Erläuterungen der Formel-Syntax. Der Zusatz **as** weist dem Feld einen neuen Namen zu.

Argument	Beschreibung
inline	Der Zusatz **inline** wird benutzt, wenn Daten direkt in das Skript eingegeben und nicht aus einer Datei geladen werden sollen. *data ::= [text]* Daten, die durch eine **inline**-Bedingung in das Skript eingefügt werden, müssen in doppelten Anführungszeichen oder in eckigen Klammern stehen. Der Text wird auf die gleiche Weise interpretiert wie der Inhalt einer Datei, d. h. er sollte auch genauso aufgebaut sein. Wenn Sie beispielsweise in einer Textdatei eine neue Zeile beginnen würden, sollten Sie dies auch im Text eines **inline**-Befehls durch Drücken der ENTER-Taste tun. Die Anzahl der Spalten wird durch die erste Zeile definiert. *format-spec ::= (fspec-item {, fspec-item })* Die Formatbezeichnung besteht aus einer Auflistung von Formatoptionen, die in Klammern stehen. Weitere Informationen finden Sie unter *Formatoptionen (page 172)*.
resident	Der Zusatz **resident** wird benutzt, um Daten aus einer bereits geladenen Tabelle zu laden. *table label* ist die Bezeichnung, die der **LOAD**-Anweisung vorangeht, durch den die Tabelle erstellt wurde. Am Ende des resident-Zusatzes sollte ein Doppelpunkt stehen.

Argument	Beschreibung
extension	Sie können Daten aus Analyseverbindungen laden. Sie müssen die **extension**-Bedingung verwenden, um eine Funktion aufzurufen, die im Plugin für serverseitige Erweiterung (SSE) definiert ist, oder ein Skript auswerten. Sie können eine einzelne Tabelle an das SSE-Plugin senden, worauf eine einzelne Datentabelle zurückgegeben wird. Wenn das Plugin nicht die Namen der zurückzugebenden Felder angibt, werden die Felder Field1, Field2 usw. benannt. `Extension pluginname.functionname( tabledescription );` • Laden von Daten mithilfe einer Funktion in einem SSE-Plugin *tabledescription ::= (table { ,tablefield})* Wenn Sie keine Tabellenfelder angeben, werden die Felder in der Ladereihenfolge verwendet. • Laden von Daten durch Auswertung eines Skripts in einem SSE-Plugin *tabledescription ::= (script, table { ,tablefield})* **Umgang mit Datentypen in der Tabellenfelddefinition** Datentypen werden in Analyseverbindungen automatisch erkannt. Wenn die Daten keinen numerischen Wert und mindestens einen Nicht-NULL-Textstring enthalten, wird das Feld als Text betrachtet. In allen anderen Fällen wird es als numerisch betrachtet. Sie können den Datentyp erzwingen, indem Sie einen Feldnamen mit **String()** oder mit **Mixed()** umgeben. • **String()** erzwingt, dass ein Feld als Text betrachtet wird. Wenn das Feld numerisch ist, wird der Textteil des dualen Werts extrahiert und keine Konvertierung vorgenommen. • **Mixed()** erzwingt, dass das Feld als dual betrachtet wird. **String()** oder **Mixed()** können nicht außerhalb von **extension**-Tabellenfelddefinitionen verwendet werden, und Sie können keine anderen Qlik Sense Funktionen in einer Tabellenfelddefinition verwenden.
where	**where** wird benutzt, um anhand eines Kriteriums zu prüfen, ob der Datensatz geladen wird oder nicht. Ist *criterion* True, wird der Datensatz geladen. *criterion* ist eine logische Formel.
while	Der Zusatz **while** dient dazu, einen Datensatz mehrfach zu laden. Der Datensatz wird solange eingelesen, wie *criterion* True ist. Sinnvollerweise schließt ein **while**-Zusatz die Funktion **IterNo( )** ein. *criterion* ist eine logische Formel.

Argument	Beschreibung
group by	Der Zusatz **group by** bestimmt, nach welchen Feldern die Daten aggregiert (gruppiert) werden sollen. Die Aggregierungsfelder sollten in den geladenen Formeln verwendet werden. Außerhalb von Aggregierungsfunktionen dürfen in load-Befehlen mit group by-Zusatz nur die im group by-Zusatz aufgeführten Felder geladen werden. *groupbyfieldlist ::= (fieldname { ,fieldname })*
order by	Der Zusatz **order by** kann nur in **load**-Befehlen mit resident-Zusatz benutzt werden, d. h. wenn Daten aus einer bereits geladenen Tabelle eingelesen werden. Er dient dazu, die Datensätze der im resident-Zusatz bezeichneten Tabelle vor dem erneuten Einlesen zu sortieren. Dabei kann nach einem oder mehreren Felder in auf- oder absteigender Reihenfolge sortiert werden. Die Sortierung erfolgt zunächst nach numerischen Werten, dann nach nationaler Sortierreihenfolge. Diese Bedingung darf nur verwendet werden, wenn die Datenquelle eine bezeichnete Tabelle ist. Mit den Ordnungsfeldern wird angegeben, nach welchem Feld die bezeichnete Tabelle sortiert wird. Das Feld kann durch den Feldnamen oder die Feldnummer bezeichnet sein (die Nummerierung beginnt stets bei 1). orderbyfieldlist ::= fieldname [sortorder] { , fieldname [sortorder] } Hinter *sortorder* folgt entweder *asc* für aufsteigend oder *desc* für absteigend. Fehlt die *sortorder*, wird *asc* angenommen. *fieldname*, *path*, *filename* und *aliasname* sind Strings. Als *fieldname* kann jedes Feld in der Quelltabelle verwendet werden. Felder, die mithilfe der as-Bedingung (*aliasname*) erstellt werden, befinden sich außerhalb des Gültigkeitsbereichs und können nicht im gleichen **load**-Befehl verwendet werden.

Let

Der **let**-Befehl ergänzt den **set**-Befehl und definiert die Skriptvariablen. Im Gegensatz zum **set**-Befehl wird beim **let**-Befehl der Ausdruck rechts des Gleichheitszeichens "=" zur Laufzeit des Skripts ausgewertet, bevor er der Variablen zugewiesen wird.

Syntax:

```
Let variablename=expression
```

Beispiele und Ergebnisse:

Beispiel	Ergebnis

`Set x=3+4;`
 	
`Let y=3+4;`
 	
`z=$(y)+1;`
	`$(x)` wird als ' 3+4 ' interpretiert
	
	`$(y)` wird als ' 7 ' interpretiert
	
	`$(z)` wird als ' 8 ' interpretiert
	
	Achten Sie auf den Unterschied zwischen den Befehlen **Set** und **Let**. Der Befehl **Set** weist den String „3+4“ zur Variablen zu, während der Befehl **Let** den String auswertet und der Variablen 7 zuweist.

Let T=now( );	`$(T)` liefert den Wert der aktuellen Uhrzeit.

Set

Der Befehl **set** wird zum Festlegen der Skriptvariablen verwendet. Diese können Strings, Pfade, Laufwerke usw. im Skript ersetzen.

Syntax:

```
Set variablename=string
```

Example 1:

```
Set FileToUse=Data1.csv;
```

Example 2:

```
Set Constant="My string";
```

Example 3:

```
Set BudgetYear=2012;
```

Put

Die **put**-Anweisung wird verwendet, um einen numerischen Wert im Hypercube festzulegen.

Der Zugriff auf die Spalten kann über Bezeichnungen erfolgen. Sie können auch nach Deklarationsreihenfolge auf Spalten und Zeilen zugreifen. Weitere Einzelheiten finden Sie in den Beispielen unten.

Syntax:

```
put column(position)=value
```

Example 1:

Der Zugriff auf die Spalten kann über Bezeichnungen erfolgen.

In diesem Beispiel wird ein Wert von `1` in der ersten Position der Spalte mit der Bezeichnung *Sales* festgelegt.

```
Put Sales(1) = 1;
```

Example 2:

Sie können auf Kennzahlenspalten in der Deklarationsreihenfolge zugreifen, indem Sie das Format `#hc1.measure` für Kennzahlen verwenden.

Dieses Beispiel legt den Wert 1000 an der zehnten Position des endgültigen sortierten Hypercubes fest.

```
Put #hc1.measure.2(10) = 1000;
```

Example 3:

Sie können auf die Dimensionszeilen in der Deklarationsreihenfolge zugreifen, indem Sie das Format `#hc1.dimension` für Dimensionen verwenden.

In diesem Beispiel wird der Wert der Konstanten Pi in die fünfte Zeile der dritten deklarierten Dimension gesetzt.

```
Put #hc1.dimension.3(5) = Pi();
```

Wenn keine entsprechenden Dimensionen oder Formeln in Werten oder Bezeichnungen vorhanden sind, wird ein Fehler zurückgegeben, der besagt, dass die Spalte nicht gefunden wurde. Wenn sich der Index für die Spalte außerhalb des Bereichs befindet, wird kein Fehler angezeigt.

HCValue

Die Funktion **HCValue** wird verwendet, um Werte in einer Zeile einer angegebenen Spalte abzurufen.

Syntax:

```
HCValue(column,position)
```

Example 1:

Dieses Beispiel gibt den Wert an der ersten Position der Spalte mit der Bezeichnung „Sales“ zurück.

```
HCValue(Sales,1)
```

Example 2:

Dieses Beispiel gibt den Wert an der zehnten Position des sortierten Hypercubes zurück.

```
HCValue(#hc1.measure2,10)
```

Example 3:

Dieses Beispiel gibt den Wert in der fünften Zeile der dritten Dimension zurück.

```
HCValue(#hc1.dimension.3,5)
```

Wenn keine entsprechenden Dimensionen oder Formeln in Werten oder Bezeichnungen vorhanden sind, wird ein Fehler zurückgegeben, der besagt, dass die Spalte nicht gefunden wurde. Wenn sich der Index für die Spalte außerhalb des Bereichs befindet, wird NULL zurückgegeben.

11 QlikView-Funktionen und -Befehle, die in Qlik Sense nicht unterstützt werden

Die meisten Funktionen und Befehle, die in Ladeskripten und Diagrammformeln in QlikView verwendet werden können, werden auch in Qlik Sense unterstützt. Die Ausnahmen werden nachfolgend beschrieben.

11.1 Skriptbefehle, die in Qlik Sense nicht unterstützt werden

QlikView Skriptbefehle, die in Qlik Sense nicht unterstützt werden

Befehl	Kommentare
Command	Stattdessen **SQL** verwenden.
InputField	

11.2 Funktionen, die in Qlik Sense nicht unterstützt werden

Diese Liste beschreibt die Skript- und Diagrammfunktionen von QlikView, die in Qlik Sense nicht unterstützt werden.

- **GetCurrentField**
- **GetExtendedProperty**
- **Input**
- **InputAvg**
- **InputSum**
- **MsgBox**
- **NoOfReports**
- **ReportComment**
- **ReportId**
- **ReportName**
- **ReportNumber**

11.3 Zusätze, die in Qlik Sense nicht unterstützt werden

Diese Liste beschreibt die Zusätze von QlikView, die in Qlik Sense nicht unterstützt werden.

- **Bundle**
- **Image_Size**
- **Info**

12 Funktionen und Befehle, die in Qlik Sense nicht empfohlen werden

Die meisten Funktionen und Befehle, die in QlikView-Ladeskripts und Diagrammformeln verwendet werden können, werden auch in Qlik Sense unterstützt. Die Verwendung einiger dieser Funktionen wird in Qlik Sense jedoch nicht empfohlen. Darüber hinaus gibt es Funktionen und Befehle, die in älteren Versionen von Qlik Sense verfügbar waren, mittlerweile aber als veraltet gekennzeichnet sind.

Aus Kompatibilitätsgründen funktionieren sie immer noch bestimmungsgemäß, aber es ist ratsam, den Code entsprechend den Empfehlungen in diesem Abschnitt zu aktualisieren, da sie in den kommenden Versionen möglicherweise entfernt werden.

12.1 Skriptbefehle, die in Qlik Sense nicht empfohlen werden

Diese Tabelle enthält Skriptbefehle, deren Verwendung in Qlik Sense nicht empfohlen wird.

Skriptbefehle, die nicht empfohlen werden

Befehl	**Empfehlung**
Command	Stattdessen **SQL** verwenden.
CustomConnect	Stattdessen **Custom Connect** verwenden.

12.2 Parameter für Skriptbefehle, die in Qlik Sense nicht empfohlen werden

Diese Tabelle beschreibt Parameter für Skriptbefehle, deren Verwendung in Qlik Sense nicht empfohlen wird.

Skriptbefehlparameter, die nicht empfohlen werden

Befehl	**Parameter**
Buffer	**Incremental** verwenden anstelle von: • **Inc** (nicht empfohlen) • **Incr** (nicht empfohlen)

Befehl	Parameter
LOAD	Die folgenden Parameterschlüsselwörter werden von QlikView-Assistenten für die Dateiumformung generiert. Die Funktion wird beim erneuten Laden der Daten beibehalten, aber Qlik Sense bietet keine geführten Support/Assistenten für die Generierung von Befehlen mit diesen Parametern: • **Bottom** • **Cellvalue** • **Col** • **Colmatch** • **Colsplit** • **Colxtr** • **Compound** • **Contain** • **Equal** • **Every** • **Expand** • **Filters** • **Intarray** • **Interpret** • **Length** • **Longer** • **Numerical** • **Pos** • **Remove** • **Rotate** • **Row** • **Rowcnd** • **Shorter**

12.3 Funktionen, die in Qlik Sense nicht empfohlen werden

Diese Tabelle beschreibt Skript- und Diagrammfunktionen, deren Verwendung in Qlik Sense nicht empfohlen wird.

Funktionen, die nicht empfohlen werden

Funktion	Empfehlung
NumAvg **NumCount** **NumMax** **NumMin** **NumSum**	Stattdessen Mengenfunktionen verwenden. *Bereichsfunktionen (page 1361)*
Color() **QliktechBlue** **QliktechGray**	Verwenden Sie stattdessen andere Farbfunktionen. **QliktechBlue()** kann durch **RGB (8, 18, 90)** und **QliktechGray** durch **RGB(158, 148, 137)** ersetzt werden, damit sich dieselben Farben erzielen lassen. *Farbfunktionen (page 572)*
QlikViewVersion	Stattdessen **EngineVersion** verwenden. *EngineVersion (page 1510)*
ProductVersion	Stattdessen **EngineVersion** verwenden. *EngineVersion (page 1510)*
QVUser	
Year2Date	Stattdessen **YearToDate** verwenden.
Vrank	Stattdessen **Rank** verwenden.
WildMatch5	Stattdessen **WildMatch** verwenden.

Qualifizierer **ALL**

In QlikView gab es den Zusatz **ALL** vor einer Formel. Dies gleicht der Verwendung von **{1} TOTAL**. Die Berechnung erfolgt hier über alle Werte des Felds im Dokument, ohne Berücksichtigung der Diagrammdimensionen und der aktuellen Auswahlen. In diesem Fall ergibt sich stets dasselbe Ergebnis, unabhängig vom logischen Status im Dokument. Der Zusatz **ALL** schließt den Gebrauch von Auswahlformeln aus, da der Zusatz **ALL** selbst eine Auswahlformel definiert. Aus Gründen der Kompatibilität wird der Zusatz **ALL** in dieser Version von Qlik Sense noch unterstützt, in späteren Versionen aber wahrscheinlich nicht mehr.