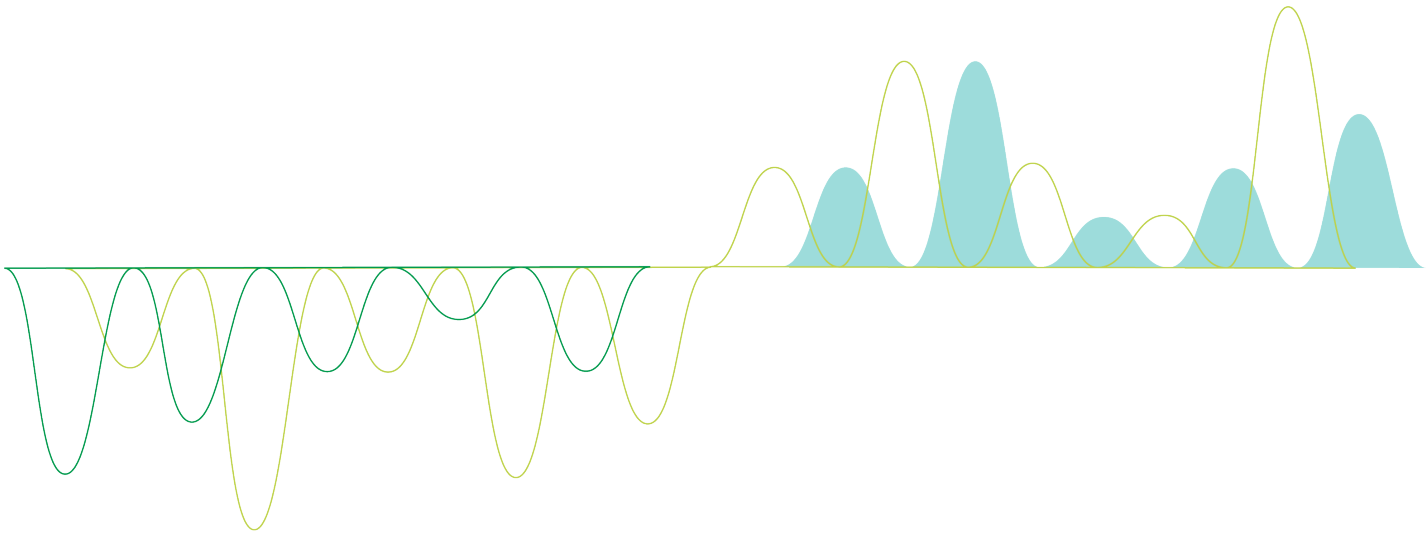
Sintaxe de scripts e funções de gráficos

Qlik Sense®
August 2023
Copyright © 1993-2023 QlikTech International AB. Todos os direitos reservados.



© 2023 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 O que é Qlik Sense? 16
1.1 O que você pode fazer no Qlik Sense? 16
1.2 Como o Qlik Sense funciona? 16
O modelo do aplicativo 16
A experiência associativa 16
Colaboração e mobilidade 16
1.3 Como você pode implantar o Qlik Sense? 16
Qlik Sense Desktop 17
Qlik Sense Enterprise 17
1.4 Como administrar e gerenciar um site do Qlik Sense 17
1.5 Estenda o Qlik Sense e adapte-o aos seus próprios objetivos 17
Criando extensões e mashups 17
Criando clientes 17
Criando ferramentas de servidor 17
Conectando a outras fontes de dados 17
2 Visão geral da sintaxe do script 18
2.1 Introdução à sintaxe do script 18
2.2 O que é formalismo Backus-Naur? 18
2 Palavras-chave e comandos de script 20
2.3 Comandos de controle de script 20
Visão geral dos comandos de controle de script 20
Call 22
Do..loop 23
End 24
Exit 25
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
2.4 Prefixos de script 34
Visão geral dos prefixos de script 34
Add 38
Buffer 39
Concatenate 41
Crosstable 47
First 56
Generic 58
Hierarchy 65
HierarchyBelongsTo 67
Inner 69
IntervalMatch 69
Join 73
Keep 83

Left 84
Mapeamento 85
Merge 87
NoConcatenate 92
Only 100
Outer 100
Carregamento parcial 101
Replace 105
Right 106
Sample 107
Semantic 110
Unless 114
When 120
2.5 Comandos regulares de script 126
Visão geral dos comandos regulares de script 126
Alias 132
AutoNumber 133
Binary 136
Comment field 138
Comment table 139
Connect 139
Declare 141
Derive 144
Direct Query 145
Directory 149
Disconnect 150
Drop 151
Drop table 152
Execute 153
Field/Fields 154
FlushLog 154
Force 154
From 156
Load 157
Let 175
Loosen Table 175
Map 176
NullAsNull 177
NullAsValue 177
Qualify 178
Rem 179
Rename 180
Search 182
Section 182
Select 183
Set 185
Sleep 186
SQL 186

SQLColumns 187
SQLTables 188
SQLTypes 188
Star 189
Store 191
Table/Tables 194
Tag 194
Trace 195
Unmap 195
Unqualify 196
Untag 196
2.6 Diretório de trabalho 197
Diretório de trabalho do Qlik Sense Desktop 197
Diretório de trabalho do Qlik Sense 198
2 Trabalhando com variáveis no editor de carregamento de dados 199
2.7 Visão geral 199
2.8 Definindo uma variável 199
2.9 Excluindo uma variável 200
2.10 Carregando um valor de variável como um valor de campo 200
2.11 Cálculo da variável 200
2.12 Variáveis de sistema 201
Visão geral das variáveis de sistema 201
CreateSearchIndexOnReload 204
HidePrefix 204
HideSuffix 205
Include 205
OpenUrlTimeout 206
StripComments 207
Verbatim 207
2.13 Variáveis de tratamento de valores 207
Visão geral das variáveis de valores manipuláveis 207
NullDisplay 208
NullInterpret 208
NullValue 209
OtherSymbol 209
2.14 Variáveis de interpretação numérica 209
Formato da moeda 210
Formato numérico 210
Formato da hora 210
BrokenWeeks 212
DateFormat 213
DayNames 219
DecimalSep 223
FirstWeekDay 226
LongDayNames 230
LongMonthNames 233
MoneyDecimalSep 237

MoneyFormat 241
MoneyThousandSep 245
MonthNames 249
NumericalAbbreviation 255
ReferenceDay 256
ThousandSep 260
TimeFormat 267
TimestampFormat 267
2.15 Variáveis do Direct Discovery 270
Variáveis de sistema do Direct Discovery 270
Variáveis de marcação da consulta do Teradata 271
Variáveisde caracteres do Direct Discovery 272
Variáveis de interpretação numérica do Direct Discovery 273
2.16 Variáveis de erro 274
Visão geral das variáveis de erro 274
ErrorMode 274
ScriptError 275
ScriptErrorCount 276
ScriptErrorList 276
2 Expressões de script 277
3 Expressões de gráfico 278
3.1 Definindo o escopo de agregação 278
3.2 Análise de conjunto 281
Expressões de conjunto 281
Exemplos 282
Conjuntos naturais 282
Identificadores de conjunto 285
Operadores de conjunto 286
Modificadores de conjunto 287
Expressões de conjunto internas e externas 308
Tutorial - Criando uma expressão de conjunto 310
Sintaxe para expressões de conjunto 320
3.3 Sintaxe geral para expressões de gráfico 320
3.4 Sintaxe geral para agregações 321
4 Operadores 322
4.1 Operadores de bit 322
4.2 Operadores lógicos 323
4.3 Operadores numéricos 323
4.4 Operadores relacionais 324
4.5 Operadores de caractere 325
& 326
like 326
5 Funções de script e gráfico 327
5.1 Conexões analíticas para extensões do lado do servidor (SSE) 327
5.2 Funções de agregação 327
Usando funções de agregação em um script de carga de dados 328

Usando funções de agregação em expressões do gráfico 328
Como agregações são calculadas 328
Agregação de campos-chave 328
Funções básicas de agregação 329
Funções de agregação de contador 351
Funções de agregação financeiras 369
Funções de agregação estatística 397
Funções estatísticas de teste 467
Funções de agregação de string 536
Funções de dimensão sintética 549
Agregações aninhadas 552
5.3 Aggr - função de gráfico 552
Exemplos: Expressões de gráfico usando Agg 555
5.4 Funções de cor 558
Funções de cores pré-definidas 560
ARGB 561
RGB 562
HSL 564
5.5 Funções condicionais 565
Visão geral das funções condicionais 565
alt 566
class 567
coalesce 568
if 569
match 573
mixmatch 576
pick 579
wildmatch 580
5.6 Funções de contador 583
Visão geral das funções de contador 583
autonumber 584
autonumberhash128 586
autonumberhash256 588
IterNo 590
RecNo 591
RowNo 592
RowNo - função de gráfico 594
5.7 Funções de data e hora 596
Visão geral de funções de data e hora 597
addmonths 605
addyears 615
age 622
converttolocaltime 624
day 628
dayend 634
daylightsaving 642
dayname 643
daynumberofquarter 645

daynumberofyear 651
daystart 658
firstworkdate 665
GMT 667
hour 671
inday 674
indaytotime 683
inlunarweek 693
inlunarweektodate 705
inmonth 716
inmonths 724
inmonthstodate 738
inmonthtodate 751
inquarter 761
inquartertodate 774
inweek 786
inweektodate 803
inyear 817
inyeartodate 829
lastworkdate 842
localtime 851
lunarweekend 855
lunarweekname 866
lunarweekstart 879
makedate 891
maketime 897
makeweekdate 904
minute 912
month 918
monthend 924
monthname 934
monthsend 942
monthsname 955
monthsstart 968
monthstart 981
networkdays 990
now 1000
quarterend 1007
quartername 1020
quarterstart 1032
second 1044
setdateyear 1049
setdateyearmonth 1051
timezone 1053
today 1054
UTC 1060
week 1060
weekday 1076

weekend 1085
weekname 1097
weekstart 1112
weekyear 1125
year 1134
yearend 1141
yearname 1153
yearstart 1165
yeartodate 1177
5.8 Funções exponenciais e logarítmicas 1193
5.9 Funções de campo 1194
Funções de contagem 1194
Funções de campo e seleção 1195
GetAlternativeCount - função de gráfico 1195
GetCurrentSelections - função de gráfico 1196
GetExcludedCount - função de gráfico 1198
GetFieldSelections - função de gráfico 1199
GetNotSelectedCount - função de gráfico 1202
GetObjectDimension - função de gráfico 1202
GetObjectField - função de gráfico 1203
GetObjectMeasure - função de gráfico 1204
GetPossibleCount - função de gráfico 1204
GetSelectedCount - função de gráfico 1206
5.10 Funções de arquivo 1207
Visão geral das funções de arquivo 1207
Attribute 1209
ConnectString 1218
FileBaseName 1218
FileDir 1219
FileExtension 1219
FileName 1219
FilePath 1220
FileSize 1220
FileTime 1221
GetFolderPath 1222
QvdCreateTime 1223
QvdFieldName 1224
QvdNoOfFields 1225
QvdNoOfRecords 1226
QvdTableName 1227
5.11 Funções financeiras 1228
Visão geral das funções financeiras 1229
BlackAndSchole 1229
FV 1230
nPer 1231
Pmt 1232
PV 1233
Rate 1234

5.12 Funções de formato ... 1235
Visão geral das funções de formatação ... 1235
ApplyCodepage ... 1236
Date ... 1237
Dual ... 1239
Interval ... 1241
Money ... 1242
Num ... 1243
Time ... 1246
Timestamp ... 1247
5.13 Funções numéricas gerais ... 1248
Visão geral das funções numéricas gerais ... 1249
Funções de combinação e permutação ... 1249
Funções modulares ... 1250
Funções de paridade ... 1250
Funções de arredondamento ... 1250
BitCount ... 1251
Ceil ... 1251
Combin ... 1252
Div ... 1253
Even ... 1253
Fabs ... 1254
Fact ... 1254
Floor ... 1255
Fmod ... 1256
Frac ... 1257
Mod ... 1257
Odd ... 1258
Permut ... 1258
Round ... 1259
Sign ... 1260
5.14 Funções geoespaciais ... 1261
Visão geral das funções geoespaciais ... 1261
GeoAggrGeometry ... 1263
GeoBoundingBox ... 1264
GeoCountVertex ... 1264
GeoGetBoundingBox ... 1265
GeoGetPolygonCenter ... 1265
GeoInvProjectGeometry ... 1266
GeoMakePoint ... 1267
GeoProject ... 1267
GeoProjectGeometry ... 1268
GeoReduceGeometry ... 1269
5.15 Funções de interpretação ... 1270
Visão geral das funções de interpretação ... 1270
Date# ... 1272
Interval# ... 1273
Money# ... 1273

Num# 1275
Text 1275
Time# 1276
Timestamp# 1277
5.16 Funções interregistro 1278
Funções de linha 1279
Funções de coluna 1279
Funções de campo 1280
Funções de tabela dinâmica 1280
Funções inter-registro no script de carga de dados 1281
Above - função de gráfico 1282
Below - função de gráfico 1286
Bottom - função de gráfico 1290
Column - função de gráfico 1294
Dimensionality - função de gráfico 1296
Exists 1297
FieldIndex 1301
FieldValue 1303
FieldValueCount 1304
LookUp 1306
NoOfRows - função de gráfico 1308
Peek 1310
Previous 1317
Top - função de gráfico 1319
SecondaryDimensionality - função de gráfico 1323
After - função de gráfico 1323
Before - função de gráfico 1325
First - função de gráfico 1326
Last - função de gráfico 1327
ColumnNo - função de gráfico 1328
NoOfColumns - função de gráfico 1328
5.17 Funções lógicas 1329
5.18 Funções de mapeamento 1330
Visão geral das funções de mapeamento 1330
ApplyMap 1331
MapSubstring 1332
5.19 Funções matemáticas 1334
5.20 Funções NULL 1335
Visão geral das funções NULL 1335
EmptyIsNull 1335
IsNull 1336
NULL 1337
5.21 Funções de intervalo 1338
Funções básicas de intervalo 1338
Funções de intervalo de contador 1339
Funções de intervalo estatístico 1339
Funções de intervalo financeiro 1340
RangeAvg 1341

RangeCorrel 1343
RangeCount 1345
RangeFractile 1347
RangeIRR 1349
RangeKurtosis 1350
RangeMax 1351
RangeMaxString 1353
RangeMin 1355
RangeMinString 1357
RangeMissingCount 1358
RangeMode 1360
RangeNPV 1362
RangeNullCount 1363
RangeNumericCount 1364
RangeOnly 1366
RangeSkew 1367
RangeStdev 1368
RangeSum 1370
RangeTextCount 1372
RangeXIRR 1373
RangeXNPV 1375
5.22 Funções relacionais 1377
Funções de classificação 1377
Funções de agrupamento 1378
Funções de decomposição de séries temporais 1379
Rank - função de gráfico 1380
HRank - função de gráfico 1384
Otimização com o k-means: Um exemplo do mundo real 1386
KMeans2D - função de gráfico 1395
KMeansND - função de gráfico 1410
KMeansCentroid2D - função de gráfico 1425
KMeansCentroidND - função de gráfico 1427
STL_Trend - função de gráfico 1428
STL_Seasonal - função de gráfico 1430
STL_Residual - função de gráfico 1432
Tutorial - Decomposição de séries temporais no Qlik Sense 1434
5.23 Funções estatísticas de distribuição 1437
Visão geral das funções de distribuição estatística 1438
BetaDensity 1440
BetaDist 1441
BetaInv 1441
BinomDist 1442
BinomFrequency 1442
BinomInv 1442
ChiDensity 1443
ChiDist 1443
ChiInv 1444
FDensity 1445

FDist ... 1445
FInv ... 1446
GammaDensity ... 1446
GammaDist ... 1447
GammaInv ... 1447
NormDist ... 1448
NormInv ... 1448
PoissonDist ... 1449
PoissonFrequency ... 1449
PoissonInv ... 1450
TDensity ... 1450
TDist ... 1451
TInv ... 1451
5.24 Funções de string ... 1452
Visão geral das funções da cadeia de caracteres ... 1452
Capitalize ... 1456
Chr ... 1456
Evaluate ... 1457
FindOneOf ... 1457
Hash128 ... 1458
Hash160 ... 1459
Hash256 ... 1460
Index ... 1461
IsJson ... 1462
JsonGet ... 1463
JsonSet ... 1464
KeepChar ... 1464
Left ... 1466
Len ... 1466
LevenshteinDist ... 1467
Lower ... 1468
LTrim ... 1469
Mid ... 1470
Ord ... 1471
PurgeChar ... 1471
Repeat ... 1472
Replace ... 1473
Right ... 1474
RTrim ... 1475
SubField ... 1476
SubStringCount ... 1479
TextBetween ... 1480
Trim ... 1481
Upper ... 1482
5.25 Funções do sistema ... 1482
Visão geral das funções do sistema ... 1482
EngineVersion ... 1485
InObject - função de gráfico ... 1485

IsPartialReload 1489
ObjectId - função de gráfico 1490
ProductVersion 1492
StateName - função de gráfico 1493
5.26 Funções de tabela 1493
Visão geral das funções da tabela 1493
FieldName 1495
FieldNumber 1496
NoOfFields 1496
NoOfRows 1497
5.27 Funções trigonométricas e hiperbólicas 1497
6 Restrição de acesso do sistema de arquivo 1500
6.1 Aspectos de segurança ao conectar-se com conexões de dados ODBC e OLE DB baseadas em arquivos 1500
6.2 Limitações do modo padrão 1500
Variáveis de sistema 1500
Comandos regulares de script 1502
Comandos de controle de script 1503
Funções de arquivo 1504
Funções do sistema 1506
6.3 Desativando o modo padrão 1506
Qlik Sense 1507
Qlik Sense Desktop 1507
6 Scripts em nível de gráfico 1508
6.4 Comandos de controle 1508
Visão geral de instruções de comando do modificador de gráfico 1508
Call 1510
Do..loop 1511
End 1512
Exit 1512
Exit script 1512
For..next 1513
For each..next 1514
If..then..elseif..else..end if 1517
Next 1518
Sub..end sub 1518
Switch..case..default..end switch 1520
To 1520
6.5 Prefixos 1521
Visão geral de prefixos de modificadores de gráfico 1521
Add 1521
Replace 1522
6.6 Comandos regulares 1522
Visão geral de comandos regulares de modificadores de gráfico 1522
Load 1523
Let 1527
Set 1528

Put ... 1528
HCValue ... 1529
7 Funções e comandos do QlikView não suportados em Qlik Sense ... 1530
7.1 Comandos de script não suportados em Qlik Sense ... 1530
7.2 Funções não suportadas em Qlik Sense ... 1530
7.3 Prefixos não suportados no Qlik Sense ... 1530
8 Funções e comandos não recomendados em Qlik Sense ... 1531
8.1 Comandos de script não recomendados em Qlik Sense ... 1531
8.2 Parâmetros de comandos de script não recomendados em Qlik Sense ... 1531
8.3 Funções não recomendadas em Qlik Sense ... 1533
Qualificador ALL ... 1533

1 O que é Qlik Sense?

O Qlik Sense é uma plataforma para a análise de dados. Com o Qlik Sense você pode analisar os dados e fazer suas próprias descobertas. Você pode compartilhar o conhecimento e a análise de dados em grupos e organizações. O Qlik Sense permite que você pergunte, responda às suas próprias perguntas e siga seus caminhos para as ideias. O Qlik Sense permite que você e seus colegas tomem decisões de forma colaborativa.

1.1 O que você pode fazer no Qlik Sense?

A maioria dos produtos de Business Intelligence (BI) pode ajudá-lo a responder às perguntas preexistentes. Mas e as perguntas de acompanhamento? Aquelas que surgem depois que alguém lê o seu relatório ou vê sua visualização? Com a experiência associativa do Qlik Sense, você pode responder pergunta após pergunta após pergunta, movendo-se ao longo do seu próprio caminho para a ideia. Com o Qlik Sense você pode explorar os dados livremente, apenas com cliques, aprendendo a cada passo do caminho e desenvolvendo novos passos com base em resultados anteriores.

1.2 Como o Qlik Sense funciona?

O Qlik Sense gera exibições das informações em tempo real para você. O Qlik Sense não exige relatórios predefinidos e estáticos ou que você dependa de outros usuários – basta clicar e aprender. Cada vez que você clica, o Qlik Sense responde instantaneamente, atualizando cada visualização no Qlik Sense com um conjunto recém calculado de dados e visualizações específicas para suas seleções.

O modelo do aplicativo

Em vez de implementar e gerenciar grandes aplicativos de negócios, você pode criar seus próprios aplicativos Qlik Sense que poderão ser reutilizados, modificados e compartilhados com outras pessoas. O modelo do aplicativo ajuda a fazer e responder à próxima pergunta em seu país, sem ter que recorrer a um especialista para um novo relatório ou visualização.

A experiência associativa

O Qlik Sense gerencia automaticamente todas as relações nos dados e apresenta informações usando uma metáfora **green/white/gray**. As seleções são destacadas em verde, os dados associados são representados em branco e os dados excluídos (não associados) aparecem em cinza. Esse feedback instantâneo permite pensar em novas perguntas e continuar a explorar e descobrir.

Colaboração e mobilidade

O Qlik Sense permite que você colabore ainda mais com seus colegas, não importa quando e onde eles estão localizados. Todos os recursos do Qlik Sense, incluindo a experiência e a colaboração associativas, estão disponíveis em dispositivos móveis. Com o Qlik Sense, você pode perguntar e responder às suas perguntas e ainda acompanhá-las com seus colegas, onde quer que esteja.

1.3 Como você pode implantar o Qlik Sense?

Existem duas versões do Qlik Sense para implantar, Qlik Sense Desktop e Qlik Sense Enterprise.

Qlik Sense Desktop

Essa é uma versão de usuário único que é fácil de instalar e normalmente instalada em um computador local.

Qlik Sense Enterprise

Essa versão é usada para implantar sites do Qlik Sense. Um site é uma coleção de uma ou mais máquinas de servidores conectadas a um repositório lógico ou nó central comum.

1.4 Como administrar e gerenciar um site do Qlik Sense

Com o Qlik Management Console, você pode configurar, gerenciar e monitorar os sites do Qlik Sense de uma forma fácil e intuitiva. Você pode gerenciar licenças, regras de acesso e segurança, configurar os nós e as conexões da fonte de dados e sincronizar o conteúdo e usuários, entre muitas outras atividades e recursos.

1.5 Estenda o Qlik Sense e adapte-o aos seus próprios objetivos

Qlik Sense fornece APIs e SDKs flexíveis para desenvolver suas próprias extensões e adaptar e integrar o Qlik Sense para diferentes objetivos, como:

Criando extensões e mashups

Aqui você pode fazer o desenvolvimento web utilizando o JavaScript para criar extensões de visualização personalizada nos aplicativos Qlik Sense, ou usar APIs de mashups para criar sites com conteúdo do Qlik Sense.

Criando clientes

Você pode criar clientes no .NET e incorporar objetos do Qlik Sense em seus próprios aplicativos. Você também pode criar clientes nativos em qualquer linguagem de programação que lide com a comunicação WebSocket usando o protocolo cliente do Qlik Sense.

Criando ferramentas de servidor

Com APIs de serviço e diretório de usuários, pode criar sua própria ferramenta para administrar e gerenciar sites do Qlik Sense.

Conectando a outras fontes de dados

Crie conectores Qlik Sense do para recuperar dados de fontes de dados personalizadas.

2 Visão geral da sintaxe do script

2.1 Introdução à sintaxe do script

Em um script são definidos os nomes da fonte de dados, das tabelas e dos campos incluídos na lógica. Além disso, os campos na definição de direitos de acesso são definidos no script. Os scripts consistem de diversos comandos que são executados consecutivamente.

As sintaxes da linha de comando e do script do Qlik Sense são descritas em uma notação denominada formalismo de Backus-Naur ou código BNF.

As primeiras linhas de código já são geradas quando é criado um novo arquivo Qlik Sense. Os valores padrão dessas variáveis de interpretação numérica derivam da configuração regional do sistema operacional.

Os scripts consistem em diversos comandos e palavras-chave que são executadas consecutivamente. Todos os comandos devem terminar com um ponto e vírgula **;**,

Você pode usar expressões e funções nos comandos **LOAD** para transformar os dados que foram carregados.

Para um arquivo de tabelas com vírgulas, guias ou pontos e vírgulas como delimitadores, o comando **LOAD** pode ser usado. Por padrão, o comando **LOAD** carregará todos os campos do arquivo.

Bancos de dados gerais podem ser acessados por meio dos conectores ODBC ou OLE DB de banco de dados. Aqui os comandos SQL padrão são usadas. A sintaxe SQL aceita variações entre os diferentes drivers ODBC.

Além disso, você pode acessar outras fontes de dados usando conectores personalizados.

2.2 O que é formalismo Backus-Naur?

As sintaxes da linha de comando e do script do Qlik Sense são descritas em uma notação denominada formalismo de Backus-Naur ou código BNF.

A seguinte tabela fornece uma lista de símbolos usados no código BNF, com uma descrição de como são interpretados:

Símbolos

Símbolo	**Descrição**
|	Lógica OR: o símbolo pode ser usado em qualquer um dos lados.
( )	Parênteses que definem a precedência: usados para estrutura a sintaxe BNF.
[ ]	Colchetes: os itens entre colchetes são opcionais.
{ }	Chaves: os itens entre chaves podem não ser repetidos ou ser repetidos mais vezes.
Símbolo	Uma categoria sintática não terminal, que: pode ser dividida em outros símbolos. Por exemplo, compostos dos símbolos acima e outros não terminais, caracteres de texto e assim por diante.

Símbolo	Descrição
::=	Marca o início de um bloco que define um símbolo.
LOAD	Um símbolo terminal que consiste em caracteres de texto. Deve ser escrito como está no script.

Todos os símbolos terminais são impressos em **bold face**. Por exemplo, "(" deve ser interpretado como um parêntese que define a precedência. Já "**(**" deve ser interpretado como um caractere a ser impresso no script.

Exemplo:

A descrição do comando alias é:

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

Isso deve ser interpretado como o caractere de texto "alias", seguido por um nome de campo arbitrário, seguido pelo caractere de texto "as", seguido por um nome de alias arbitrário. Qualquer quantidade de combinações adicionais de “fieldname as alias” pode ser fornecida, separada por vírgulas.

Os comandos a seguir estão corretos:

```
alias a as first;

alias a as first, b as second;

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

Já os comandos a seguir não estão corretos:

```
alias a as first b as second;

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

2 Palavras-chave e comandos de script

O script do Qlik Sense consiste em vários comandos. Um comando pode ser comum ou de controle. Alguns comandos podem ser precedidos por prefixos.

Comandos comuns geralmente são usados para manipular dados de uma forma ou de outra. Esses comandos podem ser escritos em qualquer quantidade de linhas no script e devem sempre ser encerrados por um ponto-e-vírgula ";".

Comandos de controle geralmente são utilizados para controlar o fluxo de execução do script. Cada cláusula de um comando de controle deve ser mantido dentro de uma linha do script e pode ser encerrada por um ponto e vírgula ou pelo fim de linha.

Os prefixos podem ser usados com comandos comuns aplicáveis, mas nunca com comandos de controle. Os prefixos **when** e **unless** podem, entretanto, ser usados como sufixos para cláusulas de comando de controle menos específicas.

No próximo subcapítulo, você encontrará uma lista alfabética de todos os comandos de script, comandos de controle e prefixos.

Todas as palavras-chave do script podem ser digitadas com qualquer combinação de caracteres maiúsculos e minúsculos. No entanto, os nomes de campos e de variáveis usados nos comandos diferenciam maiúsculas de minúsculas.

2.3 Comandos de controle de script

O script do Qlik Sense consiste em vários comandos. Um comando pode ser comum ou de controle.

Comandos de controle geralmente são utilizados para controlar o fluxo de execução do script. Cada cláusula de um comando de controle deve ser mantido dentro de uma linha do script e pode ser encerrado por ponto e vírgula ou fim da linha.

Jamais são aplicados prefixos aos comandos de controle, com exceção dos prefixos **when** e **unless**, que podem ser usados com alguns comandos de controle específicos.

Todas as palavras-chave do script podem ser digitadas com qualquer combinação de caracteres maiúsculos e minúsculos.

Visão geral dos comandos de controle de script

Cada função é descrita adicionalmente após a visão geral. Você também pode clicar no nome da função na sintaxe para acessar imediatamente os detalhes dessa função específica.

Call

O comando de controle **call** chama uma sub-rotina que deve ser definida por um comando **sub** prévio.

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

Do..loop

A declaração de controle **do..loop** é uma construção de iteração de script que executa um ou vários comandos até uma condição lógica ser atendida.

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

Exit script

Esse comando de controle interrompe a execução do script. Ele pode ser inserido em qualquer parte do script.

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

For each ..next

O comando de controle **for each..next** cria uma construção de iteração de script que executa um ou vários comandos para cada valor de uma lista separada por vírgulas. Os comandos dentro do loop incluídos entre **for** e **next** serão executados para cada valor da lista.

```
For each..next var in list

[statements]

[exit for [ ( when | unless ) condition ]

[statements]

next [var]
```

For..next

O comando de controle **for..next** cria uma construção de iteração de script com um contador. Os comandos dentro do loop incluídos entre **for** e **next** serão executados para cada valor da variável do contador, entre os limites inferior e superior especificados.

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

[statements]

[exit for [ ( when | unless ) condition ]

[statements]

Next [counter]
```

If..then

O comando de controle **if..then** é uma construção de seleção do script que força a execução do script seguir caminhos diferentes, dependendo de uma ou várias condições lógicas.

*Como o comando **if..then** é um comando de controle e, portanto, termina com um ponto e vírgula ou com um fim de linha, cada uma das quatro cláusulas possíveis (**if..then**,**elseif..then**, **else** e **end if**) não deve ultrapassar o limite da linha.*

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

  [ statements ]

{ elseif condition then

  [ statements ] }

[ else

  [ statements ] ]

end if
```

Sub

A declaração de controle **sub..end sub** define uma sub-rotina que pode ser acionada por meio de um comando **call**.

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

Switch

O comando de controle **switch** é uma construção de seleção do script que força a execução do script siga caminhos diferentes, dependendo do valor de uma expressão.

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

Call

O comando de controle **call** chama uma sub-rotina que deve ser definida por um comando **sub** prévio.

Sintaxe:

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

Argumentos:

Argumentos

Argumento	Descrição
name	O nome da sub-rotina.

Argumento	Descrição
paramlist	Uma lista separada por vírgulas dos parâmetros reais que serão enviados para a sub-rotina. Cada item da lista deve ser um nome de campo, uma variável ou uma expressão arbitrária.

A sub-rotina chamada por uma declaração **call** deve ser definida por um **sub** encontrado anteriormente durante a execução do script.

Os parâmetros são copiados na sub-rotina e, se o parâmetro no comando **call** for uma variável e não uma expressão, serão copiados novamente ao sair da sub-rotina.

Limitações:

- Como o comando **call** é de controle e, portanto, termina com um ponto e vírgula ou com um fim de linha, ele não deve cruzar um limite de linha.
- Quando você define uma sub-rotina com `Sub..end sub` dentro de um comando de controle, por exemplo `if..then`, você só pode chamar a sub-rotina de dentro do mesmo comando de controle.

Exemplo:

Este exemplo lista todos os arquivos relacionados do Qlik em uma pasta e em suas subpastas, e armazena informações sobre o arquivo em uma tabela. Supõe-se que você tenha criado uma conexão de dados com o nome Apps à pasta.

A sub-rotina DoDir é chamada com a referência à pasta, 'lib://Apps', como parâmetro. Dentro da sub-rotina, há uma chamada recursiva, `Call DoDir (Dir)`, que faz com que a função procure arquivos recursivamente em subpastas.

```
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

A declaração de controle **do..loop** é uma construção de iteração de script que executa um ou vários comandos até uma condição lógica ser atendida.

Sintaxe:

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

*Como o comando **do..loop** é um comando de controle e, portanto, termina com um ponto e vírgula ou com um fim de linha, cada uma das três cláusulas possíveis (**do**,**exit do** e **loop**) não deve ultrapassar o limite da linha.*

Argumentos:

Argumentos

Argumento	Descrição
condition	Uma expressão lógica de avaliação como True ou False.
statements	Qualquer grupo de um ou mais comandos de script do Qlik Sense.
while / until	A cláusula condicional **while** ou **until** deve aparecer apenas uma vez em qualquer declaração **do..loop**, isto é, depois de **do** ou depois de **loop**. Cada condição é interpretada somente na primeira ocorrência, mas é avaliada em todas as outras ocorrências no loop.
exit do	Se uma cláusula **exit do** for encontrada dentro do loop, a execução do script será transferida para o primeiro comando após a cláusula **loop**, indicando o fim do loop. Uma cláusula **exit do** pode ser transformada em condicional pelo uso opcional de um sufixo **when** ou **unless**.

Exemplo:

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

Set a=1;

Do while a<10

LOAD * from file$(a).csv;

Let a=a+1;

Loop
```

End

A palavra chave do script **End** é usada para fechar cláusulas **If**,**Sub** e **Switch**.

Exit

A palavra chave do script **Exit** é parte do comando **Exit Script**, mas também pode ser usada para sair das cláusulas **Do**, **For** ou **Sub**.

Exit script

Esse comando de controle interrompe a execução do script. Ele pode ser inserido em qualquer parte do script.

Sintaxe:

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

Como o comando **exit script** é de controle e, portanto, termina com um ponto e vírgula ou com um fim de linha, ele não deve cruzar um limite de linha.

Argumentos:

Argumentos

Argumento	Descrição
condition	Uma expressão lógica de avaliação como True ou False.
when / unless	Um comando **exit script** pode ser transformado em condicional pelo uso opcional de um sufixo **when** ou **unless**.

Exemplos:

```
//Exit script
Exit Script;

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

For..next

O comando de controle **for..next** cria uma construção de iteração de script com um contador. Os comandos dentro do loop incluídos entre **for** e **next** serão executados para cada valor da variável do contador, entre os limites inferior e superior especificados.

Sintaxe:

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

[statements]

[exit for [ ( when | unless ) condition ]

[statements]
```

```
Next [counter]
```

As expressões *expr1*, *expr2* e *expr3* serão avaliadas somente na primeira vez que o loop é inserido. O valor da variável counter pode ser alterado por comandos dentro da referência circular, entretanto, essa não é uma prática de programação recomendável.

Se uma cláusula **exit for** for encontrada dentro do loop, a execução do script será transferida para o primeiro comando após a cláusula **next**, indicando o fim do loop. Uma cláusula **exit for** pode ser transformada em condicional pelo uso opcional de um sufixo **when** ou **unless**.

*Como o comando **for..next** é um comando de controle e, portanto, termina com um ponto e vírgula ou com um fim de linha, cada uma das três cláusulas possíveis (**for..to..step**,**exit for** e **next**) não deve ultrapassar o limite da linha.*

Argumentos:

Argumentos

Argumento	Descrição
counter	Um nome de variável. Se *counter* for especificado depois de **next**, ele deverá ter o mesmo nome de variável que a encontrada após o **for** correspondente.
expr1	Uma expressão que determina o primeiro valor da variável *counter* para o qual o loop deve ser executado.
expr2	Uma expressão que determina o último valor da variável *counter* para o qual o loop deve ser executado.
expr3	Uma expressão que determina o valor que indica o incremento da variável *counter* cada vez que o loop é executado.
condition	uma expressão lógica de avaliação como True ou False.
statements	Qualquer grupo de um ou mais comandos de script do Qlik Sense.

Example 1: Carregando uma sequência de arquivos

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

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

Example 2: Carregando um número aleatório de arquivos

Neste exemplo, presumimos que existem os arquivos de dados *x1.csv, x3.csv, x5.csv, x7.csv* e *x9.csv*. O carregamento é interrompido em um ponto aleatório usando a condição `if rand( )<0.5 then`.

```
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

O comando de controle **for each..next** cria uma construção de iteração de script que executa um ou vários comandos para cada valor de uma lista separada por vírgulas. Os comandos dentro do loop incluídos entre **for** e **next** serão executados para cada valor da lista.

Sintaxe:
A sintaxe especial permite gerar listas com nomes de arquivo e diretório no diretório atual.

```
for each var in list

[statements]

[exit for [ ( when | unless ) condition ]

[statements]

next [var]
```

Argumentos:

Argumentos

Argumento	Descrição
var	Um nome de uma variável de script que adquire um novo valor da lista para cada execução de referência de loop. Se **var** for especificado depois de **next**, ele deverá ter o mesmo nome de variável que a encontrada após o **for each** correspondente.

O valor da variável **var** pode ser alterado por comandos dentro da referência circular, entretanto, essa não é uma prática de programação recomendável.

Se uma cláusula **exit for** for encontrada dentro do loop, a execução do script será transferida para o primeiro comando após a cláusula **next**, indicando o fim do loop. Uma cláusula **exit for** pode ser transformada em condicional pelo uso opcional de um sufixo **when** ou **unless**.

*Como o comando **for each..next** é um comando de controle e, portanto, termina com um ponto e vírgula ou com um fim de linha, cada uma das três cláusulas possíveis (**for each**,**exit for** e **next**) não deve ultrapassar o limite da linha.*

Sintaxe:

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

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

Argumentos

Argumento	Descrição
constant	Qualquer número ou string. Observe que uma string escrita diretamente no script deve ser colocada entre aspas simples. Uma string sem aspas simples será interpretada como uma variável e, em seguida, o valor da variável será usado. Os números não precisam ser colocados entre aspas simples.
expression	Uma expressão arbitrária.
mask	A máscara de um nome de arquivo ou de um nome de pasta que pode incluir todos os caracteres válidos de nome de arquivo, bem como os caracteres curingas padrão ***** e **?**. Você pode usar caminhos de arquivo absolutos ou caminhos lib://.
condition	Uma expressão lógica de avaliação como True ou False.
statements	Qualquer grupo de um ou mais comandos de script do Qlik Sense.
filelist mask	Essa sintaxe produz uma lista separada por vírgulas de todos os arquivos existentes no diretório atual e que correspondem à máscara de nome de arquivo. *Este argumento suporta apenas as conexões de dados da biblioteca no modo padrão.*
dirlist mask	Essa sintaxe produz uma lista separada por vírgulas de todas as pastas existentes na pasta atual e que correspondem à máscara de nome de pasta. *Este argumento suporta apenas as conexões de dados da biblioteca no modo padrão.*
fieldvaluelist mask	Essa sintaxe itera através dos valores de um campo já carregado no Qlik Sense.

*O Qlik Conectores do provedor de armazenamento na Web e outras conexões de DataFiles não oferecem suporte para máscaras de filtro que usam os caracteres curinga (***** e **?**).*

Example 1: Carregando uma lista de arquivos

```
// 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: Criando uma lista de arquivos no disco

Este exemplo carrega uma lista de todos os arquivos do Qlik Sense relacionados em uma pasta.

```
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: Iterações através dos valores de um campo

Este exemplo itera através da lista dos valores carregados do FIELD e gera um novo campo, NEWFIELD. Para cada valor do FIELD, será criado dois registros NEWFIELD.

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

FOR Each a in FieldValueList('FIELD')
```

```
LOAD '$(a)' &'-'&RecNo() as NEWFIELD AutoGenerate 2;
NEXT a
```

A tabela resultante tem a seguinte aparência:

Example table

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

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

O comando de controle **if..then** é uma construção de seleção do script que força a execução do script seguir caminhos diferentes, dependendo de uma ou várias condições lógicas.

Comandos de controle geralmente são utilizados para controlar o fluxo de execução do script. Em uma expressão de gráfico, use a função condicional **if** em vez disso.

Sintaxe:

```
If  condition then

  [ statements ]

{ elseif condition then

  [ statements ] }

[ else

  [ statements ] ]

end if
```

Como o comando **if..then** é um comando de controle e, portanto, termina com um ponto e vírgula ou com um fim de linha, cada uma das quatro cláusulas possíveis (**if..then**,**elseif..then**, **else** e **end if**) não deve ultrapassar o limite da linha.

Argumentos:

Argumentos

Argumento	Descrição
condition	Uma expressão lógica que pode ser avaliada como True ou False.
statements	Qualquer grupo de um ou mais comandos de script do Qlik Sense.

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

A palavra chave do script **Next** é usada para fechar loops **For**.

Sub..end sub

A declaração de controle **sub..end sub** define uma sub-rotina que pode ser acionada por meio de um comando **call**.

Sintaxe:

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

Argumentos são copiados na sub-rotina e, se os parâmetros reais correspondentes na instrução **call** forem o nome de uma variável, eles serão copiados novamente após a saída da sub-rotina.

Se uma sub-rotina tiver mais parâmetros formais que os parâmetros reais transmitidos por um comando **call**, os parâmetros extra serão inicializados como NULL e poderão ser utilizados como variáveis locais na sub-rotina.

Argumentos:

Argumentos

Argumento	Descrição
name	O nome da sub-rotina.
paramlist	Uma lista, separada por vírgulas, de nomes de variáveis para os parâmetros formais da sub-rotina. Eles podem ser usados como qualquer variável dentro da sub-rotina.
statements	Qualquer grupo de um ou mais comandos de script do Qlik Sense.

Limitações:

- Como o comando **sub** é um comando de controle e, portanto, termina com um ponto e vírgula ou com um fim de linha, cada uma de suas duas cláusulas possíveis (**sub** e **end sub**) não deve ultrapassar o limite da linha.
- Quando você define uma sub-rotina com `Sub..end sub` dentro de um comando de controle, por exemplo `if..then`, você só pode chamar a sub-rotina de dentro do mesmo comando de controle.

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: – transferência de parâmetro

```
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)
```

O resultado do exibido acima será local, dentro da sub-rotina, A será inicializado para 1, B será inicializado para 4 e C será inicializado para NULL.

Durante a saída da sub-rotina, a variável global A receberá 2 como valor (copiado da sub-rotina). O segundo parâmetro real "(X+1)*2" não será copiado de volta porque não é uma variável. Por fim, a variável global C não será afetada pela chamada de sub-rotina.

Switch..case..default..end switch

O comando de controle **switch** é uma construção de seleção do script que força a execução do script siga caminhos diferentes, dependendo do valor de uma expressão.

Sintaxe:

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

> *Como o comando **switch** é um comando de controle e, portanto, termina com um ponto e vírgula ou com um fim de linha, cada uma das quatro cláusulas possíveis (**switch**,**case**, **default** e **end switch**) não deve ultrapassar o limite da linha.*

Argumentos:

Argumentos

Argumento	Descrição
expression	Uma expressão arbitrária.
valuelist	Uma lista de valores separados por vírgulas com os quais o valor da expressão será comparado. A execução do script continuará com os comandos do primeiro grupo encontrado com um valor em valuelist igual ao valor da expressão. Cada valor da valuelist pode ser uma expressão arbitrária. Se nenhuma correspondência for encontrada em nenhuma cláusula **case**, as declarações da cláusula **default**, se especificadas, serão executadas.
statements	Qualquer grupo de um ou mais comandos de script do Qlik Sense.

Exemplo:

```
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

A palavra chave do script **To** é usada em vários comandos de script.

2.4 Prefixos de script

Os prefixos podem ser usados com comandos comuns aplicáveis, mas nunca com comandos de controle. Os prefixos **when** e **unless** podem, entretanto, ser usados como sufixos para cláusulas de comando de controle menos específicas.

Todas as palavras-chave do script podem ser digitadas com qualquer combinação de caracteres maiúsculos e minúsculos. No entanto, os nomes de campos e de variáveis usados nos comandos diferenciam maiúsculas de minúsculas.

Visão geral dos prefixos de script

Cada função é descrita adicionalmente após a visão geral. Você também pode clicar no nome da função na sintaxe para acessar imediatamente os detalhes dessa função específica.

Add
O prefixo **Add** pode ser incluído em qualquer comando **LOAD** ou **SELECT** no script para especificar que deve adicionar registros a outra tabela. Ele também especifica que esse comando deve ser executado em um carregamento parcial. O prefixo **Add** também pode ser usado em um comando **Map**.

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

Buffer
Os arquivos QVD podem ser criados e mantidos automaticamente usando o prefixo **buffer**. Esse prefixo pode ser usado com a maioria dos comandos **LOAD** e **SELECT** no script. Ele indica se os arquivos QVD serão usados para armazenar em cache/buffer o resultado do comando.

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

Concatenate
Se duas tabelas que serão concatenadas tiverem conjuntos de campos diferentes, a concatenação pode ser forçada com o prefixo **Concatenate**.

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

Crosstable

O prefixo de carregamento **crosstable** é usado para transpor dados estruturados de "tabela cruzada" ou "tabela dinâmica". Dados estruturados dessa forma são comumente encontrados ao trabalhar com fontes de planilhas. A saída e o objetivo do prefixo de carregamento **crosstable** é transpor essas estruturas para uma tabela regular equivalente orientada por colunas, já que essa estrutura geralmente é mais adequada para análises no Qlik Sense.

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

First

O prefixo **First** em um comando **LOAD** ou **SELECT (SQL)** é usado para carregar um número máximo configurado de registros de uma tabela de fonte de dados.

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

Generic

O prefixo de carregamento **Generic** permite a conversão de dados modelados de entidade/atributo/valor (EAV) em uma estrutura de tabela relacional normalizada tradicional. A modelagem de EAV é alternativamente chamada de "modelagem de dados genéricos" ou "esquema aberto".

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

Hierarchy

O prefixo **hierarchy** é usado para transformar uma tabela de hierarquia pai-filho em uma tabela útil em um modelo de dados do Qlik Sense. Ele pode ser colocado na frente de um comando **LOAD** ou **SELECT** e usará o resultado do comando de carregamento como entrada para uma transformação de tabela.

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

HierarchBelongsTo

Este prefixo é usado para transformar uma tabela de hierarquia pai-filho em uma tabela que seja útil em um modelo de dados do Qlik Sense. Ele pode ser colocado na frente de um comando **LOAD** ou **SELECT** e usará o resultado do comando de carregamento como entrada para uma transformação de tabela.

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

Inner

Os prefixos **join** e **keep** podem ser precedidos pelo prefixo **inner**.

Se usado antes de **join**, especifica que inner join deve ser usado. Por isso, a tabela resultante só conterá combinações de valores de campo das tabelas de dados brutos nas quais os valores de campo de link são representados em ambas as tabelas. Se usado antes de **keep**, especificará que as tabelas de dados brutos deverão ser reduzidas à sua interseção comum antes de serem armazenadas no Qlik Sense.

.

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

IntervalMatch

O prefixo **IntervalMatch** estendido é usado para criar uma tabela comparando valores numéricos discretos com um ou mais intervalos numéricos e opcionalmente comparar os valores de uma ou várias chaves adicionais.

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

Join

O prefixo **join** une a tabela carregada a uma tabela nomeada existente ou à última tabela de dados criada.

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

Keep

O prefixo **keep** é semelhante ao prefixo **join**. Assim como o prefixo **join**, ele compara a tabela carregada com uma tabela nomeada existente ou com a última tabela de dados criada anteriormente. No entanto, em vez de juntar a tabela carregada com uma existente, ele que tem o efeito de reduzir uma ou ambas as tabelas antes de elas serem armazenadas no Qlik Sense, com base no cruzamento de dados da tabela. A comparação feita é equivalente a uma junção natural feita em todos os campos comuns, ou seja, da mesma maneira como acontece em uma junção correspondente. Entretanto, as duas tabelas não são unidas e serão armazenadas no Qlik Sense como duas tabelas nomeadas separadamente.

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

Left

Os prefixos **Join** e **Keep** podem ser precedidos pelo prefixo **left**.

Se usado antes de **join**, especifica que left join deve ser usado. A tabela resultante só conterá combinações de valores de campo das tabelas de dados brutas nas quais os valores de campo de link são representados na primeira tabela. Se usado antes de **keep**, especificará que a segunda tabela de dados brutos deverá ser reduzida à sua interseção comum com a primeira tabela antes de ser armazenada no Qlik Sense.

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

Mapping

O prefixo **mapping** é usado para criar uma tabela de mapeamento que possa ser usada, por exemplo, para substituir valores e nomes de campo durante a execução do script.

```
Mapeamento  ( loadstatement | selectstatement )
```

Merge

O prefixo **Merge** pode ser adicionado a qualquer comando **LOAD** ou **SELECT** no script para especificar que a tabela carregada deve ser mesclada em outra tabela. Ele também especifica que esse comando deve ser executado em um carregamento parcial.

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

NoConcatenate

O prefixo **NoConcatenate** força duas tabelas carregadas com conjuntos de campos idênticos a serem tratadas como tabelas internas à parte, quando do contrário seriam concatenadas automaticamente.

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

Outer

O prefixo **Join** explícito pode ser precedido pelo prefixo **Outer** para especificar uma junção externa. Em uma junção externa, todas as combinações entre as duas tabelas são geradas. A tabela resultante conterá combinações de valores de campo das tabelas de dados brutas em que os valores de campo de ligação são representados em uma ou ambas as tabelas. A palavra-chave **Outer** é opcional e é o tipo de junção padrão usado quando um prefixo join não é especificado.

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

Partial reload

Um carregamento total sempre começa excluindo todas as tabelas no modelo de dados existente e, em seguida, executa o script de carregamento.

Um *Carregamento parcial (page 101)* não fará isso. Em vez disso, ele mantém todas as tabelas no modelo de dados e, em seguida, executa apenas os comandos **Load** e **Select** precedidos por um prefixo **Add**, **Merge** ou **Replace**. Outras tabelas de dados não são afetadas pelo comando. O argumento **only** indica que o comando deve ser executado apenas durante carregamentos parciais e deve ser ignorado durante carregamentos totais. A tabela a seguir resume a execução da instrução para carregamentos parciais e totais.

Replace

O prefixo **Replace** pode ser adicionado a qualquer comando **LOAD** ou **SELECT** no script para especificar que a tabela carregada deve substituir outra tabela. Ele também especifica que esse comando deve ser executado em um carregamento parcial. O prefixo **Replace** também pode ser usado em um comando **Map**.

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

Right

Os prefixos **Join** e **Keep** podem ser precedidos pelo prefixo **right**.

Se usado antes de **join**, especifica que right join deve ser usado. A tabela resultante só conterá combinações de valores de campo das tabelas de dados brutas nas quais os valores de campo de link são representados na segunda tabela. Se usado antes de **keep**, especificará que a primeira tabela de dados brutos deverá ser reduzida à sua interseção comum com a segunda tabela antes de ser armazenada no Qlik Sense.

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

Sample

O prefixo **sample** em um comando **LOAD** ou **SELECT** é usado para carregar uma amostra aleatória de registros de uma fonte de dados.

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

Semantic
Tabelas que contêm relações entre registros podem ser carregadas com o prefixo **semantic**. Podem ser, por exemplo, autorreferências dentro de uma tabela, na qual um registro aponta para outro, como pai, pertence a ou antecessor.

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

Unless
O prefixo e o sufixo **unless** são utilizados para criar uma cláusula condicional que determina se um comando ou uma cláusula exit deve ser avaliada. Pode ser considerado como uma alternativa compacta do comando **if..end if**.

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

When
O prefixo e o sufixo **when** são utilizados para criar uma cláusula condicional que determina se um comando ou uma cláusula exit deve ser executada. Pode ser considerado como uma alternativa compacta do comando **if..end if**.

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

Add

O prefixo **Add** pode ser incluído em qualquer comando **LOAD** ou **SELECT** no script para especificar que deve adicionar registros a outra tabela. Ele também especifica que esse comando deve ser executado em um carregamento parcial. O prefixo **Add** também pode ser usado em um comando **Map**.

Para que o carregamento parcial funcione corretamente, o aplicativo deve ser aberto com dados antes que um carregamento parcial seja disparado.

Execute um carregamento parcial usando o botão **Carregar**. Você também pode usar o Qlik Engine JSON API.

Sintaxe:

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

```
Add [only] mapstatement
```

Durante um carregamento normal (não parcial), a construção **Add LOAD** funcionará como um comando **LOAD** normal. Registros serão gerados e armazenados em uma tabela.

Se o prefixo **Concatenate** for usado, ou se existir uma tabela com o mesmo conjunto de campos, os registros serão anexados à tabela existente relevante. Caso contrário, a construção **Add LOAD** criará uma nova tabela.

Um carregamento parcial fará o mesmo. A única diferença é que a construção **Add LOAD** nunca criará uma nova tabela. Sempre existe uma tabela relevante da execução do script anterior à qual os registros devem ser anexados.

Não é feita a verificação de duplicatas. Portanto, um comando usando o prefixo **Add** geralmente incluirá um qualificador distinto ou uma cláusula where contra duplicatas.

O comando **Add Map...Using** faz com que o mapeamento ocorra também durante a execução parcial do script.

Argumentos:

Argumentos

Argumento	Descrição
only	Um qualificador opcional indicando que o comando deve ser executado somente durante carregamentos parciais. Ele deve ser desconsiderado durante carregamentos normais (não parciais).

Exemplos e resultados:

Exemplo	Resultado

`Tab1:`
 	
`LOAD Name, Number`
`FROM Persons.csv;`
 	
`Add LOAD Name, Number`
`FROM newPersons.csv;`
	Durante uma recarga normal, os dados são carregados de *Persons.csv* e armazenados na tabela Tab1 do Qlik Sense. Os dados de *NewPersons.csv* são concatenados com a mesma tabela do Qlik Sense.
	
	Durante uma recarga parcial, os dados são carregados de *NewPersons.csv* e anexados à tabela Tab1 do Qlik Sense. Não é feita a verificação de duplicatas.

`Tab1:`
 	
`SQL SELECT Name,`
`Number FROM`
`Persons.csv;`
 	
`Add LOAD Name, Number`
`FROM NewPersons.csv`
`where not exists`
`(Name);`
	É realizada uma checagem de duplicatas verificando se Name existe na tabela de dados carregada anteriormente.
	
	Durante uma recarga normal, os dados são carregados de *Persons.csv* e armazenados na tabela Tab1 do Qlik Sense. Os dados de *NewPersons.csv* são concatenados com a mesma tabela do Qlik Sense.
	
	Durante uma recarga parcial, os dados são carregados de *NewPersons.csv* e anexados à tabela Tab1 do Qlik Sense. É realizada uma busca por duplicatas por meio da verificação da existência de Name nos dados da tabela carregada anteriormente.

`Tab1:`
 	
`LOAD Name, Number`
`FROM Persons.csv;`
 	
`Add Only LOAD Name,`
`Number FROM`
`NewPersons.csv where`
`not exists(Name);`
	Durante uma recarga normal, os dados são carregados de *Persons.csv* e armazenados na tabela Tab1 do Qlik Sense. O comando que carrega *NewPersons.csv* é ignorado.
	
	Durante uma recarga parcial, os dados são carregados de *NewPersons.csv* e anexados à tabela Tab1 do Qlik Sense. É realizada uma busca por duplicatas por meio da verificação da existência de Name nos dados da tabela carregada anteriormente.

Buffer

Os arquivos QVD podem ser criados e mantidos automaticamente usando o prefixo **buffer**. Esse prefixo pode ser usado com a maioria dos comandos **LOAD** e **SELECT** no script. Ele indica se os arquivos QVD serão usados para armazenar em cache/buffer o resultado do comando.

Sintaxe:

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

Se não for usada nenhuma opção, o buffer de QVD criado pela primeira execução do script será usado indefinidamente.

O arquivo de buffer é armazenado na subpasta *Buffers*, em geral *C:\ProgramData\Qlik\Sense\Engine\Buffers* (instalação do servidor) ou *C:\Users\{user}\Documents\Qlik\Sense\Buffers* (Qlik Sense Desktop).

O nome do arquivo QVD é um nome calculado, um hash hexadecimal de 160 bits do seguinte comando **LOAD** ou **SELECT** inteiro e outras informações distintas. Isso significa que o buffer QVD buffer será invalidado por qualquer alteração no comando **LOAD** ou **SELECT**.

Os buffers de QVD normalmente serão excluídos quando deixarem de ser referenciados durante a execução completa do script no aplicativo que os criou ou quando o aplicativo que os criou não existir mais.

Argumentos:

Argumentos

Argumento	Descrição
incremental	A opção incremental permite a leitura apenas de parte de um arquivo subjacente. O tamanho anterior do arquivo é armazenado no cabeçalho XML no arquivo QVD. Isso é especificamente útil com arquivos de log. Todos os registros carregados anteriormente são lidos no arquivo QVD, ao passo que os novos registros subsequentes são lidos na fonte original e, por fim, é criado um arquivo QVD atualizado. A opção incremental só pode ser usada com instruções de **LOAD** e arquivos de texto. O carregamento incremental não pode ser usado quando dados antigos são alterados ou excluídos.
stale [after] amount [(days | hours)]	amount é um número que especifica o período de tempo. Os decimais podem ser utilizados. A unidade adotada será dias, se for omitida. A opção stale after é geralmente usada com fontes de DB nas quais não há nenhum carimbo de data/hora simples nos dados originais. Especifica-se por quanto tempo os snapshots capturados no último QVD poderão ser utilizados. Uma cláusula stale after simplesmente determina um período de tempo a partir da hora de criação do buffer de QVD, após a qual o buffer deixará de ser considerado válido. Antes desse período, o buffer de QVD será usado como fonte dos dados e, após o período, será usada a fonte de dados original. O arquivo do buffer de QVD será automaticamente atualizado e um novo período será iniciado.

Limitações:

Existem várias limitações. A mais evidente é que deve haver uma declaração **LOAD** ou **SELECT** de arquivo na base de qualquer declaração complexa.

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` é um prefixo de carregamento de script que permite que um conjunto de dados seja anexado a uma tabela já existente na memória. Geralmente, ele é usado para acrescentar diferentes conjuntos de dados transacionais a uma única tabela central de fatos ou para criar conjuntos de dados de referência comuns de um tipo específico que se originam de várias fontes. Sua funcionalidade é semelhante a um operador SQL UNION.

A tabela resultante de uma operação `concatenate` conterá o conjunto de dados original com as novas linhas de dados anexadas à parte inferior dessa tabela. As tabelas de origem e de destino podem ter campos diferentes presentes. Nos pontos em que os campos forem diferentes, a tabela resultante será ampliada para representar o resultado combinado de todos os campos presentes na tabela de origem e na tabela de destino.

Sintaxe:

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

Argumentos

Argumento	Descrição
`tablename`	O nome de uma tabela existente. A tabela nomeada será o destino da operação `Concatenate`, e todos os registros de dados carregados serão anexados a essa tabela. Se o parâmetro `tablename` não for usado, a tabela de destino será a última tabela carregada antes dessa instrução.
`loadstatement/selectstatement`	O argumento `loadstatement/selectstatement` que segue o argumento `tablename` será concatenado à tabela especificada.

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplo de função

Exemplo	Resultado
`Concatenate (Transactions) Load …. ;`	Os dados carregados na instrução de carregamento abaixo do prefixo `Concatenate` serão anexados à tabela existente na memória, denominada `Transactions`, (supondo que uma tabela denominada `Transactions` tenha sido carregada antes desse ponto no script de carregamento.

Exemplo 1: Anexando vários conjuntos de dados a uma tabela de destino com o prefixo de carregamento Concatenate

Script de carregamento e resultados

Visão geral

Neste exemplo, você carregará dois scripts em ordem sequencial.

- O primeiro script de carregamento contém um conjunto de dados inicial com datas e valores que é enviado para uma tabela denominada `Transactions`.
- O segundo script de carregamento contém:
    - Um segundo conjunto de dados que é anexado ao conjunto de dados inicial usando o prefixo `Concatenate`. Esse conjunto de dados tem um campo adicional, `type`, que não está no conjunto de dados inicial.
    - O prefixo `Concatenate`.

Abra o editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia.

Primeiro script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- id
- date
- amount

Primeira tabela de resultados do script de carregamento

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

A tabela mostra o conjunto de dados inicial.

Segundo script de carregamento

Abra o editor de carregamento de dados e adicione o script de carregamento abaixo.

```
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
];
```

Resultados

Carregue os dados e acesse a pasta. Crie esse campo como uma dimensão:

- type

Segunda tabela de resultados do script de carregamento

id	date	amount	type
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	-
3758	10/01/2018	164.27	Interno
3759	10/03/2018	384.00	Externo
3760	10/06/2018	25.82	Interno
3761	10/09/2018	312.00	Interno
3762	10/15/2018	4.56	Interno
3763	10/16/2018	90.24	Interno
3764	10/18/2018	19.32	Externo

Observe os valores nulos no campo `type` dos primeiros sete registros carregados em que `type` não foi definido.

Exemplo 2: Anexar vários conjuntos de dados a uma tabela de destino usando a concatenação implícita

Script de carregamento e resultados

Visão geral

Um caso de uso típico para anexar dados implicitamente é quando você carrega vários arquivos de dados estruturados de maneira idêntica e deseja anexar todos eles a uma tabela de destino.

Por exemplo, usando `wildcards` em nomes de arquivos com uma sintaxe como:

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

ou em loops usando construções como:

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

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

```
next file
```

A concatenação implícita ocorrerá entre quaisquer duas tabelas carregadas com campos com nomes idênticos, mesmo que não estejam definidos um após o outro no script. Isso pode fazer com que dados sejam anexados involuntariamente a tabelas. Se não quiser que uma tabela secundária com campos idênticos seja anexada dessa forma, use o prefixo de carregamento `NoConcatenate`*. Renomear a tabela com uma tag de nome de tabela alternativa não é suficiente para evitar que a concatenação implícita ocorra. Para obter mais informações, consulte NoConcatenate (page 92).*

Neste exemplo, você carregará dois scripts em ordem sequencial.

- O primeiro script de carregamento contém um conjunto de dados inicial com quatro campos que é enviado para uma tabela denominada `Transactions`.
- O segundo script de carregamento contém um conjunto de dados com os mesmos campos do primeiro conjunto de dados.

Abra o editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia.

Primeiro script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

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

Primeira tabela de resultados do script de carregamento

id	date	type	amount
3758	10/01/2018	Interno	164.27
3759	10/03/2018	Externo	384.00

id	date	type	amount
3760	10/06/2018	Interno	25.82
3761	10/09/2018	Interno	312.00
3762	10/15/2018	Interno	4.56
3763	10/16/2018	Interno	90.24
3764	10/18/2018	Externo	19.32

A tabela mostra o conjunto de dados inicial.

Segundo script de carregamento

Abra o editor de carregamento de dados e adicione o script de carregamento abaixo.

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

Resultados

Carregue os dados e acesse a pasta.

Segunda tabela de resultados do script de carregamento

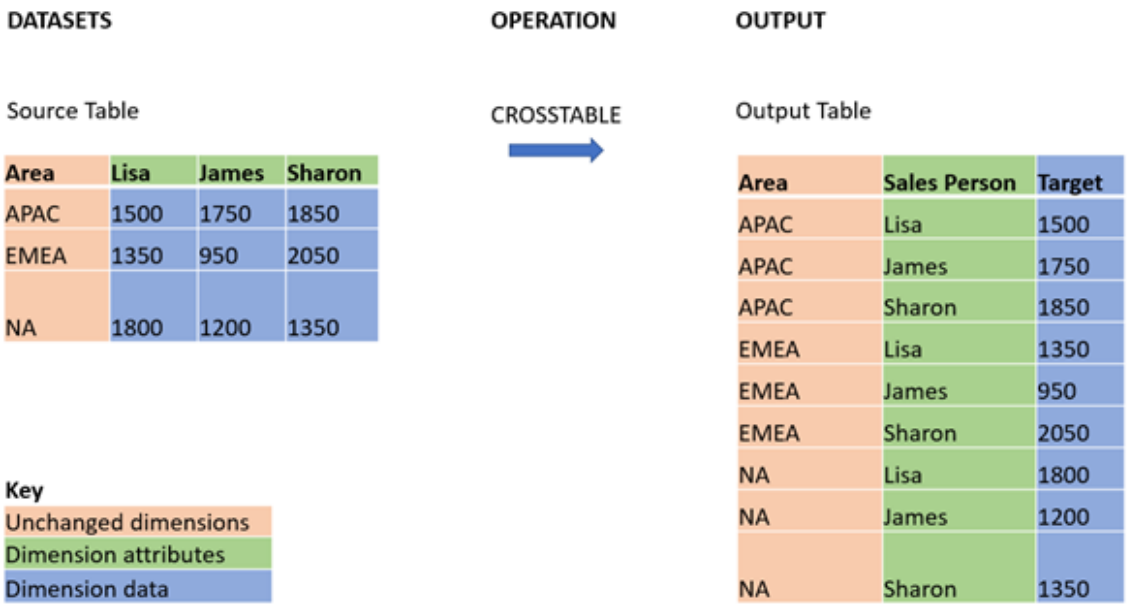
id	date	type	amount
3758	10/01/2018	Interno	164.27
3759	10/03/2018	Externo	384.00
3760	10/06/2018	Interno	25.82
3761	10/09/2018	Interno	312.00
3762	10/15/2018	Interno	4.56
3763	10/16/2018	Interno	90.24
3764	10/18/2018	Externo	19.32
3765	11/03/2018	Interno	129.40
3766	11/05/2018	Externo	638.50

O segundo conjunto de dados foi implicitamente concatenado no conjunto de dados inicial porque tinham campos idênticos.

Crosstable

O prefixo de carregamento **crosstable** é usado para transpor dados estruturados de "tabela cruzada" ou "tabela dinâmica". Dados estruturados dessa forma são comumente encontrados ao trabalhar com fontes de planilhas. A saída e o objetivo do prefixo de carregamento **crosstable** é transpor essas estruturas para uma tabela regular equivalente orientada por colunas, já que essa estrutura geralmente é mais adequada para análises no Qlik Sense.

Exemplo de dados estruturados como uma tabela cruzada e sua estrutura equivalente após uma transformação de tabela cruzada



Sintaxe:

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

Argumentos

Argumento	Descrição
attribute field name	O nome do campo de saída desejado que descreve a dimensão horizontalmente orientada a ser transposta (a linha do cabeçalho).
data field name	O nome do campo de saída desejado que descreve os dados horizontalmente orientados da dimensão a ser transposta (a matriz de valores de dados abaixo da linha do cabeçalho).
n	O número de campos de qualificador, ou dimensões inalteradas, que precedem a tabela a ser transformada em um formato genérico. O valor padrão é 1.

Essa função de script está relacionada às seguintes funções:

Funções relacionadas

Função	Interação
Generic (page 58)	Um prefixo de carregamento de transformação que usa um conjunto de dados estruturados de entidade/atributo/valor e o transforma em uma estrutura de tabela relacional regular, separando cada atributo encontrado em um novo campo ou coluna de dados.

Exemplo 1: Transformando dados de vendas pivotados (simples)

Scripts de carregamento e resultados

Visão geral

Abra o Editor de carregamento de dados e adicione o primeiro script de carregamento abaixo a uma nova guia.

O primeiro script de carregamento contém um conjunto de dados ao qual o prefixo do script `crosstable` será aplicado posteriormente, com a seção aplicando `crosstable` assinalado como comentário. Isso significa que a sintaxe de comentário foi usada para desativar essa seção no script de carregamento.

O segundo script de carregamento é igual ao primeiro, mas com a aplicação de `crosstable` sem barras de comentário (o que é possível pela remoção da sintaxe de comentário). Os scripts são mostrados dessa forma para destacar o valor dessa função de script na transformação dos dados.

Primeiro script de carregamento (função não aplicada)

```
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;
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `Product`
- `Jan 2021`

- Feb 2021
- Mar 2021
- Apr 2021
- May 2021
- Jun 2021

Tabela de resultados

Product	Jan 2021	Fev 2021	Mar 2021	Abr 2021	Maio 2021	Jun 2021
A	100	98	103	63	108	82
B	284	279	297	305	294	292
C	50	53	50	54	49	51

Esse script permite a criação de uma tabela cruzada com uma coluna para cada mês e uma linha por produto. Em seu formato atual, esses dados não são fáceis de analisar. Seria muito melhor ter todos os números em um único campo e todos os meses em outro, em uma tabela de três colunas. A próxima seção explica como fazer essa transformação para a tabela cruzada.

Segundo script de carregamento (função aplicada)

Remova as barras de comentário do script removendo //. O script de carregamento deve ter a seguinte aparência:

```
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;
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- Product
- Month
- Sales

Tabela de resultados

Product	Month	Sales
A	Jan 2021	100
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

Depois que o prefixo do script for aplicado, a tabela cruzada é transformada em uma tabela estática com uma coluna para `Month` e outra para `Sales`. Isso melhora a legibilidade dos dados.

Exemplo 2: Transformação de dados pivotados de metas de vendas em uma estrutura de tabela vertical (intermediária)

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados que é carregado em uma tabela denominada Targets.
- O prefixo de carregamento `crosstable`, que transpõe os nomes dos vendedores pivotados para um

campo próprio, rotulado `Sales Person`.

- Os dados da meta de vendas associados, que são estruturados em um campo chamado `Target`.

Script de carregamento

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

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `Area`
- `Sales Person`

Adicione esta medida:

```
=Sum(Target)
```

Tabela de resultados

Área	Representante de vendas	=Sum(Target)
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

Se quiser replicar a exibição de dados como a tabela de entrada pivotada, você poderá criar uma tabela dinâmica equivalente em uma pasta.

Faça o seguinte:

1. Copie e cole a tabela que você acabou de criar na pasta.
2. Arraste o objeto de gráfico de **Tabela dinâmica** para cima da cópia da tabela recém-criada. Selecione **Converter**.
3. Clique em ✔ **Edição concluída**.
4. Arraste o campo `Sales Person` da divisória de coluna vertical até a divisória de coluna horizontal.

A tabela a seguir mostra os dados em sua forma de tabela inicial, conforme exibido no Qlik Sense:

Tabela de resultados originais, conforme mostrado no Qlik Sense

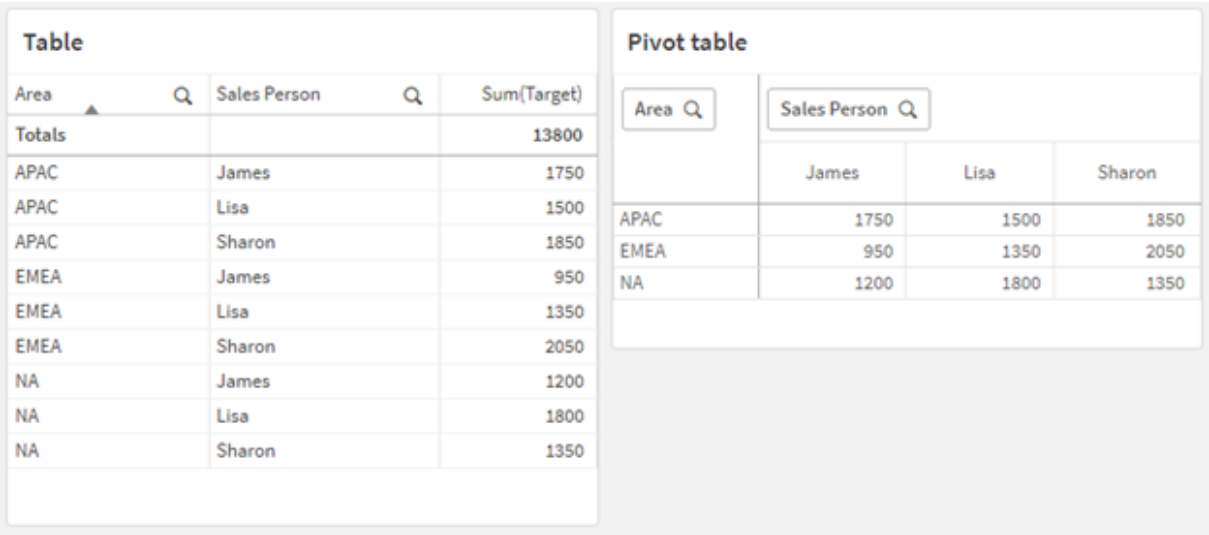
Área	Representante de vendas	=Sum(Target)
Totais	-	13800
APAC	James	1750
APAC	Lisa	1500
APAC	Sharon	1850
EMEA	James	950
EMEA	Lisa	1350
EMEA	Sharon	2050
NA	James	1200
NA	Lisa	1800
NA	Sharon	1350

A tabela dinâmica equivalente é semelhante ao seguinte, com a coluna do nome de cada vendedor contida na linha maior para `Sales Person`:

Tabela dinâmica equivalente com o campo `Sales Person` pivotado na horizontal

Área	James	Lisa	Sharon
APAC	1750	1500	1850
EMEA	950	1350	2050
NA	1350	1350	1350

Exemplo de dados exibidos como tabela e uma tabela dinâmica equivalente com o campo `Sales Person` pivotado horizontalmente



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

Exemplo 3: Transformação de dados pivotados de vendas e metas em uma estrutura de tabela vertical (avançada)

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados que representa dados de vendas e metas, organizados por área e mês do ano. Isso é carregado em uma tabela chamada `SalesAndTargets`.
- O prefixo de carregamento `crosstable`. Isso é usado para despivotar a dimensão `Month Year` em um campo dedicado, bem como para transpor a matriz de valores de vendas e metas para um campo dedicado denominado `Amount`.
- Uma conversão do campo `Month Year` de texto em uma data adequada, usando a função de conversão de texto em data `date#`. Esse campo `Month Year` convertido em data é unido novamente à tabela `SalesAndTarget` por meio de um prefixo de carregamento `Join`.

Script de carregamento

```
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;
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `Area`
- `Month Year`

Crie a seguinte medida, com o rótulo `Actual`:

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

Crie também essa medida, com o rótulo `Target`:

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

Tabela de resultados (cortada)

Área	Mês Ano	Real	Destino
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

Área	Mês Ano	Real	Destino
APAC	Dec-22	397	425
EMEA	Jan-22	363.5	362.5
EMEA	Feb-22	359.5	362.5

Se quiser replicar a exibição de dados como a tabela de entrada pivotada, você poderá criar uma tabela dinâmica equivalente em uma pasta.

Faça o seguinte:

1. Copie e cole a tabela que você acabou de criar na pasta.
2. Arraste o objeto de gráfico de **Tabela dinâmica** para cima da cópia da tabela recém-criada. Selecione **Converter**.
3. Clique em ✔ **Edição concluída**.
4. Arraste o campo `Month Year` da divisória de coluna vertical até a divisória de coluna horizontal.
5. Arraste o item `Values` da divisória de coluna horizontal até a divisória de coluna vertical.

A tabela a seguir mostra os dados em sua forma de tabela inicial, conforme exibido no Qlik Sense:

Tabela de resultados originais (cortada), conforme mostrado no Qlik Sense

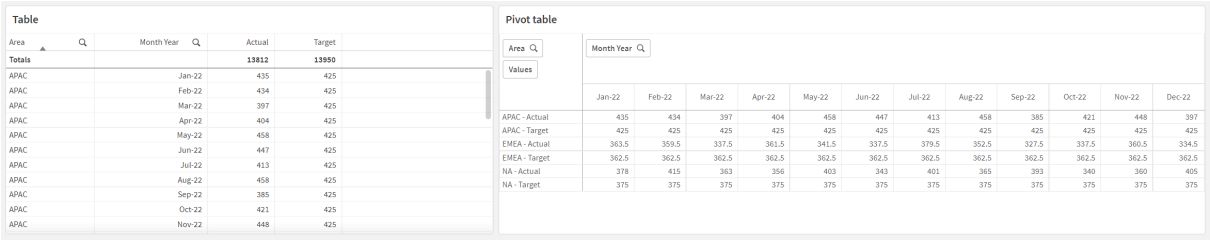
Área	Mês Ano	Real	Destino
Totais	-	13812	13950
APAC	Jan-22	435	425
APAC	Feb-22	434	425
APAC	Mar-22	397	425
APAC	Apr-22	404	425
APAC	May-22	458	425
APAC	Jun-22	447	425
APAC	Jul-22	413	425
APAC	Aug-22	458	425
APAC	Sep-22	385	425
APAC	Oct-22	421	425
APAC	Nov-22	448	425
APAC	Dec-22	397	425
EMEA	Jan-22	363.5	362.5
EMEA	Feb-22	359.5	362.5

A tabela dinâmica equivalente é semelhante à seguinte, com a coluna para cada mês individual do ano contida na linha maior para `Month Year`:

Tabela dinâmica equivalente (cortada) com o campo `Month Year` pivotado horizontalmente

Área (valores)	**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 - Real	435	434	397	404	458	447	413	458	385	421	448	397
APAC - Alvo	425	425	425	425	425	425	425	425	425	425	425	425
EMEA - Real	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 - Alvo	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 - Real	378	415	363	356	403	343	401	365	393	340	360	405
NA - Alvo	375	375	375	375	375	375	375	375	375	375	375	375

Exemplo de dados exibidos como tabela e uma tabela dinâmica equivalente com o campo `Month Year` pivotado horizontalmente



First

O prefixo `First` em uma instrução LOAD ou SELECT (SQL) é usado para carregar um número máximo definido de registros de uma tabela de fonte de dados. Um caso de uso típico para usar o prefixo `First` é quando você deseja recuperar um pequeno subconjunto de registros de uma etapa de carregamento de dados grande e/ou lenta. Assim que o número definido de registros "n" for carregado, a etapa de carregamento será encerrada prematuramente, e o restante da execução do script continuará normalmente.

Sintaxe:

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

Argumentos

Argumento	Descrição
n	Uma expressão arbitrária que é avaliada como um inteiro indicando o número máximo de registros a serem lidos. n também pode ser colocado entre parênteses: (n).
loadstatement | selectstatement	O load statement/select statement que segue o argumento n definirá a tabela especificada que deve ser carregada com o número máximo definido de registros.

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução SET DateFormat no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplos de funções

Exemplo	Resultado
FIRST 10 LOAD * from abc.csv;	Este exemplo recuperará as primeiras dez linhas de um arquivo do Excel.
FIRST (1) SQL SELECT * from Orders;	Este exemplo recuperará a primeira linha selecionada do conjunto de dados Orders.

Exemplo: Carregar as primeiras cinco linhas

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados de datas das duas primeiras semanas de 2020.
- A variável First, que instrui o aplicativo a carregar somente os cinco primeiros registros.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione `Date` como um campo e `sum(sales)` como uma medida:

Tabela de resultados

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

O script carrega apenas os cinco primeiros registros da tabela `Sales`.

Generic

O prefixo de carregamento **Generic** permite a conversão de dados modelados de entidade/atributo/valor (EAV) em uma estrutura de tabela relacional normalizada tradicional. A modelagem de EAV é alternativamente chamada de "modelagem de dados genéricos" ou "esquema aberto".

Exemplo de dados modelados por EAV e uma tabela relacional desnormalizada equivalente

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

Exemplo de dados modelados por EAV e um conjunto equivalente de tabelas relacionais normalizadas

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

Embora seja tecnicamente possível carregar e analisar dados modelados de EAV no Qlik, muitas vezes é mais fácil trabalhar com uma estrutura de dados relacional tradicional equivalente.

Sintaxe:

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

Estes tópicos podem ajudar você a trabalhar com essa função:

Tópicos relacionados

Tópico	Descrição
Crosstable (page 47)	O prefixo de carregamento `Crosstable` transforma dados orientados horizontalmente em dados orientados verticalmente. De uma perspectiva puramente funcional, ele executa a transformação oposta ao prefixo de carregamento `Generic`, embora os prefixos normalmente atendam a casos de uso totalmente diferentes.
Bancos de dados genéricos em *Gerenciar dados*	Modelos de dados estruturados de EAV são descritos com mais detalhes aqui.

Exemplo 1: Transformação de dados estruturados de EAV com o prefixo de carregamento genérico

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém um conjunto de dados que é carregado em uma tabela denominada `Transactions`. O conjunto de dados inclui um campo de data. A definição padrão de `MonthNames` é usada.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão: `Color`.

Adicione esta medida:

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

Agora, você pode inspecionar o número de produtos por cor.

Tabela de resultados

Cor	**=Count([Product ID])**
Marrom	4
Branco	2

Observe a forma do modelo de dados, em que cada atributo foi dividido em uma tabela separada, nomeada de acordo com a tag original da tabela de destino `Product`. Cada tabela tem o atributo como um sufixo. Um exemplo disso é `Product.Color`. Os registros de saída do atributo de Produto resultantes são associados pelo `Product ID`.

Representação do visualizador do modelo de dados dos resultados

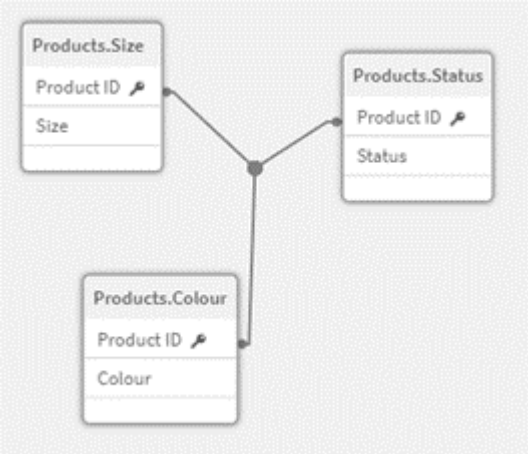


Tabela de registros resultante: Products.Status

ID do produto	Status
13	Descontinuado
2	Descontinuado

Tabela de registros resultante: Products.Size

ID do produto	Tamanho
13	13-15
20	16-18
45	16-18

Tabela de registros resultante: Products.Color

ID do produto	Cor
13	Marrom
5	Marrom
44	Marrom
45	Marrom

ID do produto	Cor
20	Branco
2	Branco

Exemplo 2: Análise de dados estruturados de EAV sem o prefixo de carregamento genérico

Script de carregamento e expressão de gráfico

Visão geral

Este exemplo mostra como analisar dados estruturados de EAV em seu formato original.

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém um conjunto de dados que é carregado em uma tabela denominada `Products` em uma estrutura de EAV.

Neste exemplo, ainda estamos contando produtos por atributo de cor. Para analisar os dados estruturados dessa maneira, você precisará aplicar a filtragem em nível de expressão dos produtos que carregam o valor do Atributo `Color`.

Além disso, atributos individuais não estão disponíveis para seleção como dimensões ou campos, dificultando determinar como criar visualizações eficazes.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão: `Value`.

Crie a seguinte medida:

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

Agora, você pode inspecionar o número de produtos por cor.

Tabela de registros resultante: Products.Status

Valor	=Count({<Attribute={'Color'}>} [Product ID])
Marrom	4
Branco	2

Exemplo 3: Desnormalização das tabelas de saída resultantes de uma carga genérica (avançada)

Script de carregamento e expressão de gráfico

Visão geral

Neste exemplo, mostramos como a estrutura de dados normalizada produzida pelo prefixo de carregamento `Generic` pode ser desnormalizada de volta em uma tabela de dimensões `Product` consolidada. Essa é uma técnica de modelagem avançada que pode ser empregada como parte do ajuste de desempenho do modelo de dados.

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

Script de carregamento

```
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;
```

Resultados

Abra o Visualizador do modelo de dados e observe a forma do modelo de dados resultante. Somente uma tabela desnormalizada está presente. É uma combinação das três tabelas de saída intermediárias: `Products.Size`, `Products.Status` e `Products.Color`.

Modelo de dados interno resultante

Products
ID do produto
Status
Cor
Tamanho

Tabela de registros resultante: Products

ID do produto	**Status**	**Cor**	**Tamanho**
13	Descontinuado	Marrom	13-15
20	-	Branco	16-18
2	Descontinuado	Branco	-
5	-	Marrom	-
44	-	Marrom	-
45	-	Marrom	16-18

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão: `Color`.

Adicione esta medida:

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

Tabela de resultados

Cor	**=Count([Product ID])**
Marrom	4
Branco	2

Hierarchy

O prefixo **hierarchy** é usado para transformar uma tabela de hierarquia pai-filho em uma tabela útil em um modelo de dados do Qlik Sense. Ele pode ser colocado na frente de um comando **LOAD** ou **SELECT** e usará o resultado do comando de carregamento como entrada para uma transformação de tabela.

O prefixo cria uma tabela de nós expandidos, que normalmente tem o mesmo número de registros como a tabela de entrada, mas cada nível na hierarquia é armazenado em um campo à parte. O campo do caminho pode ser usado em uma lista em árvore.

Sintaxe:

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

A tabela de entrada deve ser uma tabela de nós adjacentes. As tabelas de nós adjacentes são tabelas em que cada registro corresponde a um nó e tem um campo que contém uma referência para o nó pai. Em uma tabela como essa, o nó é armazenado em apenas um registro, mas o nó ainda pode ter um número indefinido de filhos. A tabela pode conter campos adicionais que descrevem atributos para os nós.

O prefixo cria uma tabela de nós expandidos, que normalmente tem o mesmo número de registros como a tabela de entrada, mas cada nível na hierarquia é armazenado em um campo à parte. O campo do caminho pode ser usado em uma lista em árvore.

Normalmente, a tabela de entrada possui exatamente um registro por nó e, nesse caso, a tabela de saída conterá o mesmo número de registros. Contudo, em alguns casos há nós com vários pais, ou seja, um nó é representado por diversos registros na tabela de entrada. Nesse caso, a tabela de saída pode ter mais registros do que a tabela de entrada.

Todos os nós com um ID pai não encontrado na coluna ID do nó (inclusive nós com ID pai não encontrado) serão considerados raiz. Além disso, somente os nós com uma conexão com um nó raiz, direta ou indireta, serão carregados, evitando-se referências circulares.

Campos adicionais que contêm o nome do nó pai, o caminho do nó e a profundidade do nó podem ser criados.

Argumentos:

Argumentos

Argumento	Descrição
NodeID	O nome do campo que contém a id do nó. O campo deve existir na tabela de entrada.
ParentID	O nome do campo que contém a id do nó do nó pai. O campo deve existir na tabela de entrada.

Argumento	Descrição
NodeName	O nome do campo que contém o nome do nó. O campo deve existir na tabela de entrada.
ParentName	Uma string usada para atribuir um nome ao novo campo **ParentName**. Se omitido, esse campo não será criado.
ParentSource	O nome do campo que contém o nome do nó usado para criar o caminho do nó. Parâmetro opcional. Se omitido, **NodeName** será usado.
PathName	Uma string usada para nomear o novo campo **Path**, que contém o caminho da raiz até o nó. Parâmetro opcional. Se omitido, esse campo não será criado.
PathDelimiter	Uma string usada como delimitador no novo campo **Path**. Parâmetro opcional. Se omitido, '/' será usado.
Depth	Uma string usada para atribuir um nome ao novo campo **Depth**, que contém a profundidade do nó na hierarquia. Parâmetro opcional. Se omitido, esse campo não será criado.

Exemplo:

```
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

Este prefixo é usado para transformar uma tabela de hierarquia pai-filho em uma tabela que seja útil em um modelo de dados do Qlik Sense. Ele pode ser colocado na frente de um comando **LOAD** ou **SELECT** e usará o resultado do comando de carregamento como entrada para uma transformação de tabela.

O prefixo cria uma tabela que contém todas as relações entre filho e ancestral da hierarquia. Então, os campos de ancestral podem ser usados para selecionar árvores inteiras da hierarquia. A tabela de saída quase sempre contém vários registros por nó.

Sintaxe:

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

A tabela de entrada deve ser uma tabela de nós adjacentes. As tabelas de nós adjacentes são tabelas em que cada registro corresponde a um nó e tem um campo que contém uma referência para o nó pai. Em uma tabela como essa, o nó é armazenado em apenas um registro, mas o nó ainda pode ter um número indefinido de filhos. A tabela pode conter campos adicionais que descrevem atributos para os nós.

O prefixo cria uma tabela que contém todas as relações entre filho e ancestral da hierarquia. Então, os campos de ancestral podem ser usados para selecionar árvores inteiras da hierarquia. A tabela de saída quase sempre contém vários registros por nó.

Um campo adicional que contém a diferença de profundidade dos nós pode ser criado.

Argumentos:

Argumentos

Argumento	Descrição
NodeID	O nome do campo que contém a id do nó. O campo deve existir na tabela de entrada.
ParentID	O nome do campo que contém a id do nó do nó pai. O campo deve existir na tabela de entrada.
NodeName	O nome do campo que contém o nome do nó. O campo deve existir na tabela de entrada.
AncestorID	Uma string usada para atribuir um nome a um novo campo da id do ancestral, que contém a id do nó do ancestral.
AncestorName	Uma string usada para atribuir um nome a um novo campo de ancestral, que contém o nome do nó do ancestral.

Argumento	Descrição
DepthDiff	Uma string usada para atribuir um nome ao novo campo **DepthDiff**, que contém o nó da hierarquia referente ao nó do ancestral. Parâmetro opcional. Se omitido, esse campo não será criado.

Exemplo:

```
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
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

Os prefixos **join** e **keep** podem ser precedidos pelo prefixo **inner**. Se usado antes de **join**, especifica que inner join deve ser usado. Por isso, a tabela resultante só conterá combinações de valores de campo das tabelas de dados brutos nas quais os valores de campo de link são representados em ambas as tabelas. Se usado antes de **keep**, especificará que as tabelas de dados brutos deverão ser reduzidas à sua interseção comum antes de serem armazenadas no Qlik Sense.

Sintaxe:

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

Argumentos:

Argumentos

Argumento	Descrição
tablename	A tabela nomeada a ser comparada com a tabela carregada.
loadstatementou selectstatement	O comando **LOAD** ou **SELECT** da tabela carregada.

Exemplo

Script de carregamento

Adicione o script de exemplo ao seu aplicativo e execute-o. Para ver o resultado, adicione os campos listados na coluna de resultados a uma pasta no seu aplicativo.

```
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 ];
```

Resultado

Tabela resultante

Column1	Column2	Column3
A	B	C
1	aa	xx

Explicação

Este exemplo demonstra a saída de Junção interna em que apenas os valores presentes na primeira (esquerda) e na segunda (direita) tabelas são unidos.

IntervalMatch

O prefixo **IntervalMatch** estendido é usado para criar uma tabela comparando valores numéricos discretos com um ou mais intervalos numéricos e opcionalmente comparar os valores de uma ou várias chaves adicionais.

Sintaxe:

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

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

O prefixo **IntervalMatch** deve ser colocado antes de um comando **LOAD** ou **SELECT** que carregue os intervalos. O campo que contém os pontos de dados discretos (Tempo, no exemplo abaixo) já deve ter sido carregado no Qlik Sense antes do comando com o prefixo **IntervalMatch**. O prefixo não lê esse campo por meio da tabela do banco de dados. O prefixo transforma a tabela carregada de intervalos e chaves em uma tabela que contém uma coluna adicional: os pontos de dados numéricos discretos. Ele também expande o número de registros de forma que a nova tabela tenha um registro por combinação possível de ponto de dados discreto, intervalo e valor do(s) campo(s) chave.

Os intervalos podem estar sobrepostos e os valores discretos estarão vinculados a todos os intervalos correspondentes.

Quando o prefixo IntervalMatch é estendido com campos chave, ele é usado para criar uma tabela comparando valores numéricos discretos com um ou mais intervalos numéricos, enquanto ao mesmo tempo compara os valores de uma ou várias chaves adicionais.

Para evitar que limites de intervalo indefinidos sejam desconsiderados, você deve permitir que valores NULL sejam mapeados para outros campos que constituem o limite inferior ou superior no intervalo. Isso pode ser controlado pelo comando **NullAsValue** ou por um teste explícito que substitui NULL por um valor numérico bem antes ou depois de qualquer um dos pontos de dados numéricos discretos.

Argumentos:

Argumentos

Argumento	Descrição
matchfield	O campo que contém os valores numéricos discretos a serem vinculados a intervalos.
keyfield	Campos que contêm os atributos adicionais a serem combinados na transformação.
loadstatement orselectstatement	Deve resultar em uma tabela cujo primeiro campo contém o limite inferior de cada intervalo, o segundo contém o limite superior e, no caso do uso da correspondência de chaves, o terceiro e quaisquer campos seguintes contêm o(s) campo(s) chave presentes no comando **IntervalMatch**. Os intervalos estão sempre fechados, isto é, sempre contêm pontos de extremidade. Os limites não numéricos fazem com que o intervalo seja desconsiderado (indefinido).

Example 1:

Nas duas tabelas abaixo, a primeira lista vários eventos discretos e a segunda define as horas inicial e final da produção de pedidos diferentes. Por meio do prefixo **IntervalMatch**, é possível conectar as duas tabelas logicamente para saber, por exemplo, quais pedidos foram afetados por algum contratempo, quais foram processados e em que turnos.

```
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;
```

Agora, a tabela **OrderLog** contém uma coluna adicional: *Time*. O número de registros também é expandido.

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: (usando keyfield)

Como no exemplo acima, adicionando *ProductionLine* como um campo chave.

```
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;
```

Agora, uma tabela poderia ser criada:

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

O prefixo **join** une a tabela carregada a uma tabela nomeada existente ou à última tabela de dados criada.

O efeito da união de dados é estender a tabela de destino por um conjunto adicional de campos ou atributos, ou seja, aqueles que ainda não estão presentes na tabela de destino. Qualquer nome de campo comum entre o conjunto de dados de origem e a tabela de destino é usado para descobrir como associar os novos registros de entrada. Isso costuma ser chamado de "união natural". Uma operação de união da Qlik pode fazer com que a tabela de destino resultante tenha mais ou menos registros do que quando começou, dependendo da exclusividade da associação de união e do tipo de união empregado.

Existem quatro tipos de uniões:

União esquerda

As uniões esquerdas são o tipo de união mais comum. Por exemplo, se você tivesse um conjunto de dados de transação e quisesse combiná-lo com um conjunto de dados de referência, normalmente usaria um `Left Join`. Você carregaria a tabela de transações primeiro e, em seguida, carregaria o conjunto de dados de referência enquanto o juntasse por meio de um prefixo `Left Join` à tabela de transações já carregada. `Left Join` manteria todas as transações como estão e adicionaria os campos de dados de referência suplementares nos quais uma correspondência fosse encontrada.

União interna

Quando houver dois conjuntos de dados nos quais você apenas se preocupa com os resultados em que há uma associação correspondente, considere usar `Inner Join`. Isso eliminará todos os registros dos dados de origem carregados e da tabela de destino se nenhuma correspondência for encontrada. Como resultado, isso

pode deixar sua tabela de destino com menos registros do que antes da operação de união.

União externa

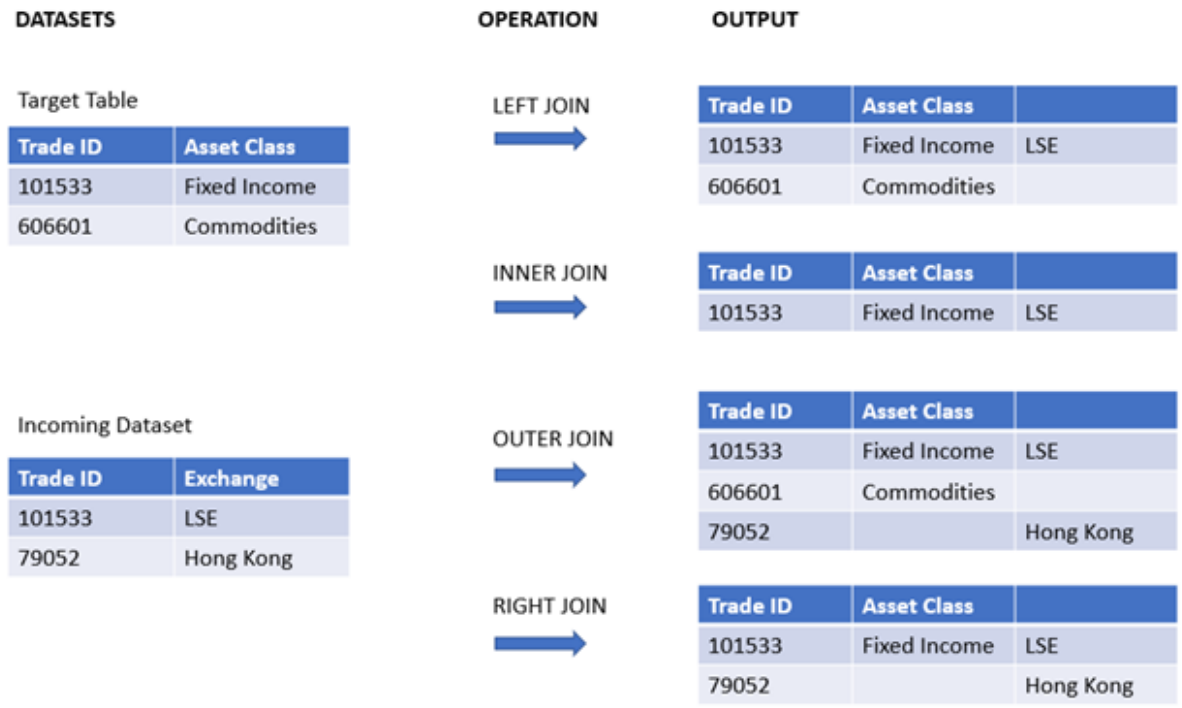
Quando você precisar manter os registros de destino e todos os registros de entrada, use `Outer Join`. Onde nenhuma correspondência for encontrada, cada conjunto de registros ainda será mantido, enquanto os campos no lado oposto da união permanecerão não preenchidos (nulos).

Se a palavra-chave "type" for omitida, o tipo de união padrão será uma união externa.

União direita

Esse tipo de união mantém todos os registros prestes a serem carregados, ao mesmo tempo em que reduz os registros na tabela direcionada pela união para somente os registros em que há uma correspondência de união nos registros de entrada. Esse é um tipo de união de nicho que às vezes é usado como meio de reduzir uma tabela de registros já pré-carregada para um subconjunto necessário.

Exemplos de conjuntos de resultados de diferentes tipos de operações de união



Se não houver nomes de campos em comum entre a origem e o destino de uma operação de união, a união resultará em um produto cartesiano de todas as linhas: isso é chamado de "união cruzada".

Exemplo de conjunto de resultados de uma operação de "união cruzada"

Sintaxe:

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

Argumentos

Argumento	Descrição
tablename	A tabela nomeada a ser comparada com a tabela carregada.
loadstatementou selectstatement	O comando **LOAD** ou **SELECT** da tabela carregada.

Estes tópicos podem ajudar você a trabalhar com essa função:

Tópicos relacionados

Tópico	Descrição
Combinando tabelas com Unir e Manter em *Gerenciar dados*	Este tópico fornece uma explicação adicional dos conceitos de "união" e "manutenção" de conjuntos de dados.
Keep (page 83)	O prefixo de carregamento `Keep` é semelhante ao prefixo `Join`, mas não combina os conjuntos de dados de origem e de destino. Em vez disso, ele apara cada conjunto de dados de acordo com o tipo de operação adotada (interna, externa, esquerda ou direita).

Exemplo 1 – União esquerda: Enriquecendo uma tabela de destino com um conjunto de dados de referência

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados que representa registros de alterações, que é carregado em uma tabela denominada `Changes`. Ele inclui um campo de chave de ID de status.
- Um segundo conjunto de dados que representa estados de alterações, que é carregado e combinado com os registros de alterações originais por meio de sua união com um prefixo de carregamento `Join` esquerdo.

Essa união esquerda garante que os registros de alterações permaneçam intactos ao adicionar atributos de status em que uma correspondência nos registros de status de entrada é encontrada com base em um ID de status comum.

Script de carregamento

```
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');
```

Resultados

Abra o Visualizador do modelo de dados e observe a forma do modelo de dados. Somente uma tabela desnormalizada está presente. É uma combinação de todos os registros de alterações originais, com os atributos de status correspondentes unidos a cada registro de alterações.

Modelo de dados interno resultante

Changes
Change ID
Status ID
Scheduled Start Date
Scheduled End Date
Business Impact
Status
Sub Status

Se você expandir a janela de visualização no Visualizador do modelo de dados, verá uma parte desse conjunto completo de resultados organizada em uma tabela:

Visualização da tabela Changes no visualizador do modelo de dados

Change ID	**Status ID**	**Scheduled Start Date**	**Scheduled End Date**	**Business Impact**	**Status**	**Sub Status**
10030	4	19/01/2022	23/02/2022	Nenhum	Closed	Cancelado
10031	3	20/01/2022	25/03/2022	Baixo	Closed	Concluído
10015	3	04/01/2022	15/02/2022	Baixo	Closed	Concluído
10103	1	02/04/2022	29/05/2022	Médio	Open	Não iniciado
10116	1	15/04/2022	24/04/2022	Nenhum	Open	Não iniciado
10134	1	03/05/2022	08/07/2022	Baixo	Open	Não iniciado
10264	1	10/09/2022	17/10/2022	Médio	Open	Não iniciado
10040	1	29/01/2022	22/04/2022	Nenhum	Open	Não iniciado
10323	1	08/11/2022	26/11/2022	Alto	Open	Não iniciado
10187	2	25/06/2022	24/08/2022	Baixo	Open	Iniciado
10185	2	23/06/2022	08/09/2022	Nenhum	Open	Iniciado

Change ID	Status ID	Scheduled Start Date	Scheduled End Date	Business Impact	Status	Sub Status
10220	2	28/07/2022	06/09/2022	Nenhum	Open	Iniciado
10326	2	11/11/2022	05/12/2022	Nenhum	Open	Iniciado
10138	2	07/05/2022	03/08/2022	Nenhum	Open	Iniciado
10334	2	19/11/2022	06/02/2023	Baixo	Open	Iniciado

Como a quinta linha da tabela Status (ID do status: '5', Status: 'Fechado', Substatus: 'Obsoleto') não corresponde a nenhum dos registros da tabela Alterações, as informações nessa linha não aparecem no conjunto de resultados acima.

Retorne ao Editor de carregamento de dados. Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão: `Status`.

Adicione esta medida:

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

Agora, você pode inspecionar o número de alterações por status.

Tabela de resultados

Status	=Count([Change ID])
Open	12
Closed	3

Exemplo 2 – União interna: Combinando somente registros correspondentes

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados que representa registros de alterações, que é carregado em uma tabela denominada `Changes`.
- Um segundo conjunto de dados que representa os registros de alterações gerados no sistema de origem `JIRA`. Esse é carregado e combinado com os registros originais por meio de sua união com um prefixo de carregamento `Inner Join`.

Essa `Inner Join` garante que apenas os cinco registros de alterações encontrados em ambos os conjuntos de dados sejam mantidos.

Script de carregamento

```
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');
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

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

Agora, você pode inspecionar os cinco registros resultantes. A tabela resultante de um `Inner Join` incluirá apenas registros com informações correspondentes em ambos os conjuntos de dados.

Tabela de resultados

Sistema de origem	Change ID	Business Impact
JIRA	10030	Nenhum
JIRA	10134	Baixo
JIRA	10220	Nenhum
JIRA	10323	Alto
JIRA	10334	Baixo

Exemplo 3 – União externa: Combinando conjuntos de registros sobrepostos

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados que representa registros de alterações, que é carregado em uma tabela denominada `Changes`.
- Um segundo conjunto de dados que representa os registros de alterações gerados no sistema de origem `JIRA`, que é carregado e combinado com os registros originais por meio de sua união com um prefixo de carregamento `Outer Join`.

Isso garante que todos os registros de alterações sobrepostos de ambos os conjuntos de dados sejam mantidos.

Script de carregamento

```
// 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');
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

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

Agora, você pode inspecionar os 10 registros resultantes.

Tabela de resultados

Sistema de origem	Change ID	Business Impact
JIRA	10030	Nenhum
JIRA	10134	Baixo
JIRA	10220	Nenhum
JIRA	10323	-
JIRA	10334	Baixo
JIRA	10597	-
-	10015	Baixo
-	10031	Baixo
-	10040	Nenhum
-	10138	Nenhum

Exemplo 4 – União direita: Aparando uma tabela de destino por um conjunto de dados mestre secundário

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados que representa registros de alterações, que é carregado em uma tabela denominada `Changes`.
- Um segundo conjunto de dados representando registros de alterações originados do sistema de origem `Teamwork`. Isso é carregado e combinado com os registros originais, juntando-o a um prefixo de carregamento `Right Join`.

Isso garante que somente os registros de alterações `Teamwork` sejam mantidos, sem perder nenhum registro `Teamwork` caso a tabela de destino não tenha um `Change ID` correspondente.

Script de carregamento

```
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');
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

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

Agora, você pode inspecionar os cinco registros resultantes.

Tabela de resultados

Sistema de origem	Change ID	Business Impact
Trabalho em equipe	10015	Baixo
Trabalho em equipe	10031	Baixo
Trabalho em equipe	10040	Nenhum
Trabalho em equipe	10103	Médio
Trabalho em equipe	50231	-

Keep

O prefixo **keep** é semelhante ao prefixo **join**. Assim como o prefixo **join**, ele compara a tabela carregada com uma tabela nomeada existente ou com a última tabela de dados criada anteriormente. No entanto, em vez de juntar a tabela carregada com uma existente, ele que tem o efeito de reduzir uma ou ambas as tabelas antes de elas serem armazenadas no Qlik Sense, com base no cruzamento de dados da tabela. A comparação feita é equivalente a uma junção natural feita em todos os campos comuns, ou seja, da mesma maneira como acontece em uma junção correspondente. Entretanto, as duas tabelas não são unidas e serão armazenadas no Qlik Sense como duas tabelas nomeadas separadamente.

Sintaxe:

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

O prefixo **keep** deve ser precedido dos prefixos **inner**, **left** ou **right**.

O prefixo **join** explícito na linguagem de script do Qlik Sense executa uma junção completa das duas tabelas. O resultado é uma tabela. Em muitos casos, essas junções resultam tabelas muito grandes. Uma das principais características do Qlik Sense é a capacidade de fazer associações entre várias tabelas, em vez de uni-las, reduzindo bastante o uso da memória, aumentando a velocidade de processamento e oferecendo

grande flexibilidade. Portanto, geralmente as junções explícitas devem ser evitadas nos scripts do Qlik Sense scripts. A funcionalidade keep foi desenvolvida para diminuir o número de casos em que é necessário usar junções explícitas.

Argumentos:

Argumentos

Argumento	Descrição
tablename	A tabela nomeada a ser comparada com a tabela carregada.
loadstatementou selectstatement	O comando **LOAD** ou **SELECT** da tabela carregada.

Exemplo:

```
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

Os prefixos **Join** e **Keep** podem ser precedidos pelo prefixo **left**.

Se usado antes de **join**, especifica que left join deve ser usado. A tabela resultante só conterá combinações de valores de campo das tabelas de dados brutas nas quais os valores de campo de link são representados na primeira tabela. Se usado antes de **keep**, especificará que a segunda tabela de dados brutos deverá ser reduzida à sua interseção comum com a primeira tabela antes de ser armazenada no Qlik Sense.

Você estava procurando pela função de caracteres pelo mesmo nome? Consulte: Left (page 1466)

Sintaxe:

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

Argumentos:

Argumentos

Argumento	Descrição
tablename	A tabela nomeada a ser comparada com a tabela carregada.
loadstatementou selectstatement	O comando **LOAD** ou **SELECT** da tabela carregada.

Exemplo

Script de carregamento

Adicione o script de exemplo ao seu aplicativo e execute-o. Para ver o resultado, adicione os campos listados na coluna de resultados a uma pasta no seu aplicativo.

```
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 ];
```

Resultado

Tabela resultante

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

Explicação

Este exemplo demonstra a saída de Junção esquerda em que somente os valores presentes na primeira tabela (esquerda) são unidos.

Mapeamento

O prefixo **mapping** é usado para criar uma tabela de mapeamento que possa ser usada, por exemplo, para substituir valores e nomes de campo durante a execução do script.

Sintaxe:

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

O prefixo **mapping** pode ser colocado na frente de uma declaração **LOAD** ou **SELECT** e usará o resultado da declaração de carregamento como entrada para uma transformação de tabela. O mapeamento fornece uma maneira eficiente de substituir valores de campo durante a execução do script, por exemplo, substituindo EUA, E.U.A ou América por EUA. A tabela de mapeamento consiste em duas colunas: a primeira contém valores de comparação, e a segunda, os valores de mapeamento desejados. As tabelas de mapeamento serão armazenadas temporariamente na memória e descartadas automaticamente após a execução do script.

O conteúdo da tabela de mapeamento pode ser acessado usando, por exemplo, o comando **Map ... Using**, o comando **Rename Field**, a função **Applymap()** ou a função **Mapsubstring()**.

Exemplo:

Neste exemplo, carregamos uma lista de vendedores com um código de país representando seu país de residência. A tabela é usada para mapear o código de um país a um país para substituir o código pelo nome do país. Somente três países estão definidos na tabela de mapeamento, outros códigos de países estão mapeados para 'Rest of the world'.

```
// 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';
```

A tabela resultante tem a seguinte aparência:

Mapping table

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

Merge

O prefixo **Merge** pode ser adicionado a qualquer comando **LOAD** ou **SELECT** no script para especificar que a tabela carregada deve ser mesclada em outra tabela. Ele também especifica que esse comando deve ser executado em um carregamento parcial.

O caso de uso típico é quando você carrega um log de alterações e deseja usá-lo para aplicar inserts, updates e deletes a uma tabela existente.

Para que o carregamento parcial funcione corretamente, o aplicativo deve ser aberto com dados antes que um carregamento parcial seja disparado.

Execute um carregamento parcial usando o botão **Carregar**. Você também pode usar o Qlik Engine JSON API.

Sintaxe:

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

Argumentos:

Argumentos

Argumento	Descrição
only	Um qualificador opcional indicando que a instrução deve ser executada somente durante carregamentos parciais. A instrução é desconsiderada durante carregamentos normais (não parciais).
SequenceNoField	O nome do campo que contém um carimbo de data/hora ou um número de sequência que define a ordem das operações.
SequenceNoVar	O nome da variável que recebe o valor máximo para SequenceNoField da tabela que está sendo mesclada.
ListOfKeys	Uma lista separada por vírgulas de nomes de campo especificando a chave primária.
Operation	O primeiro campo da instrução de Load deve conter a operação como uma string de texto: "Insert'', Update'' ou "Delete|. "i", "u" e "d" também são aceitos.

Funcionalidade geral

Durante um carregamento normal (não parcial), a construção **Merge LOAD** funcionará como uma instrução de **Load** normal, mas com a funcionalidade adicional de remover registros antigos obsoletos e registros marcados para exclusão. O primeiro campo da instrução de **Load** deve conter informações sobre a operação: Insert, Update ou Delete.

Para cada registro carregado, o identificador de registro é comparado com registros carregados anteriormente, e somente o registro mais recente (de acordo com o número de sequência) será mantido. Se o registro mais recente estiver marcado com Delete, nada será mantido.

Tabela de destino

A tabela a ser modificada é determinada pelo conjunto de campos. Se uma tabela com o mesmo conjunto de campos (exceto o primeiro campo; a operação) já existir, esta será a tabela relevante a ser modificada. Como alternativa, um prefixo **Concatenate** pode ser usado para especificar a tabela. Se a tabela de destino não for determinada, o resultado da construção **Merge LOAD** será armazenado em uma nova tabela.

Se o prefixo Concatenate for usado, a tabela resultante terá um conjunto de campos correspondentes à união da tabela existente e a entrada para a mesclagem. Portanto, a tabela de destino pode ter mais campos do que o log de alterações usado como entrada para a mesclagem.

Um carregamento parcial faz o mesmo que um carregamento total. Uma diferença é que um carregamento parcial raramente cria uma nova tabela. A menos que você tenha usado a cláusula **Only**, sempre existe uma tabela de destino com o mesmo conjunto de campos da execução do script anterior.

Número de sequência

Se o log de alterações carregado for um log acumulado, ou seja, contiver alterações que já foram carregadas, o parâmetro SequenceNoVar poderá ser usado em uma cláusula **Where** para limitar a quantidade de dados de entrada. Dessa forma, **Merge LOAD** pode ser feito para carregar somente registros em que o campo SequenceNoField é maior que SequenceNoVar. Após a conclusão, **Merge LOAD** atribui um novo valor ao SequenceNoVar com o valor máximo visto no campo SequenceNoField.

Operações

Merge LOAD pode ter menos campos do que a tabela de destino. As diferentes operações tratam os campos ausentes de maneiras diferentes:

Insert: Os campos ausentes em **Merge LOAD**, mas existentes na tabela de destino, recebem um NULL na tabela de destino.

Delete: Os campos ausentes não afetam o resultado. Os registros relevantes são excluídos de qualquer maneira.

Update: Os campos listados em **Merge LOAD** são atualizados na tabela de destino. Os campos ausentes não são alterados. Isso significa que as duas instruções a seguir não são idênticas:

- 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 ...;

A primeira instrução atualiza os registros listados e altera F2 para NULL. A segunda não muda F2, mas deixa os valores na tabela de destino.

Exemplos

Exemplo 1. Mesclagem simples com tabela especificada

Nesse exemplo, uma tabela inline chamada Persons é carregada com três linhas. Em seguida, **Merge** altera a tabela da seguinte forma:

- Adiciona a linha *Mary, 4*.
- Exclui a linha, *Steven, 3*.
- Atribui o número *5* a *Jake*.

A variável *LastChangeDate* é definida como o valor máximo na coluna *ChangeDate* após a execução de **Merge**.

Script de carregamento

Adicione o script de exemplo ao seu aplicativo e execute-o. Para ver o resultado, adicione os campos listados na coluna de resultados a uma pasta no seu aplicativo.

```
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
];
```

Resultado

Antes de **Merge Load**, a tabela resultante aparece da seguinte forma:

Resulting table

Name	Number
Jake	3
Jill	2
Steven	3

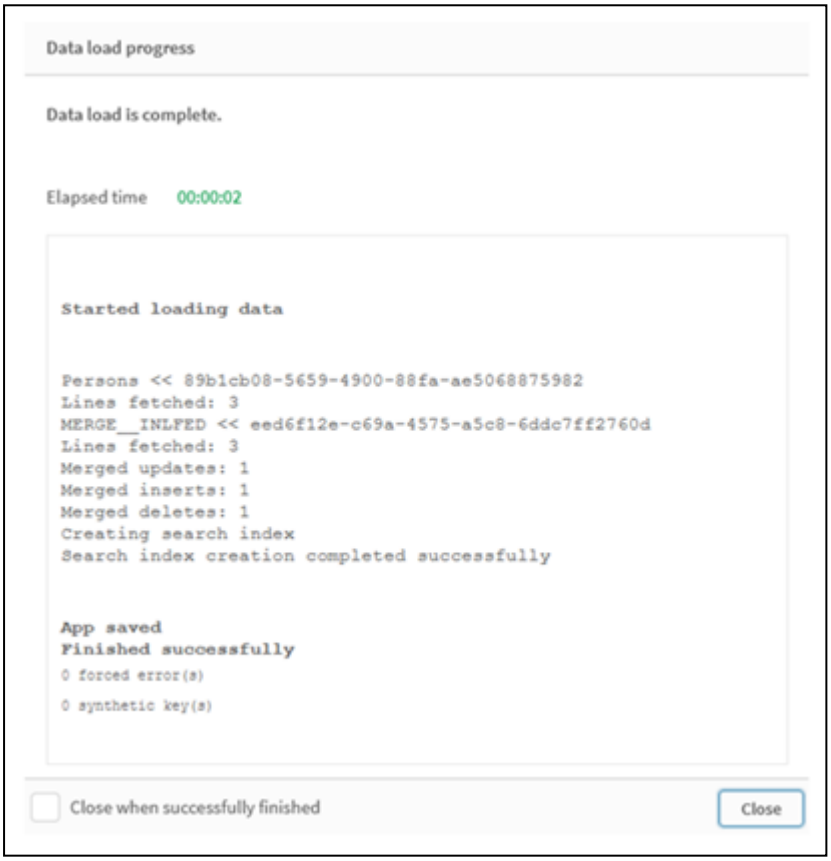
Após **Merge Load**, a tabela aparece da seguinte forma:

Resulting table

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

Quando os dados são carregados, a caixa de diálogo **Progresso da carga de dados** mostra as operações que são executadas:

Caixa de diálogo Progresso do carregamento de dados



Exemplo 2: Script de carregamento de dados com campos ausentes

Nesse exemplo, os mesmos dados acima são carregados, mas agora com um ID para cada pessoa.

Merge altera a tabela da seguinte forma:

- Adiciona a linha *Mary, 4*.
- Exclui a linha, *Steven, 3*.
- Atribui o número *5* a *Jake*.
- Atribui o número *6* a *Jill*.

Script de carregamento

Aqui, usamos duas instruções **Merge Load**, uma para "Insert" e "Delete" e uma segunda para "Update".

Adicione o script de exemplo ao seu aplicativo e execute-o. Para ver o resultado, adicione os campos listados na coluna de resultados a uma pasta no seu aplicativo.

```
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
];
```

Resultado

Após as instruções **Merge Load**, a tabela aparece da seguinte forma:

Resulting table

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

Observe que a segunda instrução **Merge** não inclui o campo **Name** e, como consequência, os nomes não foram alterados.

Exemplo 3: Script de carregamento de dados - Carregamento parcial usando uma cláusula Where com ChangeDate

No exemplo a seguir, o argumento **Only** especifica que o comando **Merge** apenas é executado durante um carregamento parcial. As atualizações são filtradas com base no LastChangeDate capturado anteriormente. Após a conclusão de **Merge**, a variável LastChangeDate recebe o valor máximo da coluna ChangeDate processada durante a mesclagem.

Script de carregamento

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

NoConcatenate

O prefixo **NoConcatenate** força duas tabelas carregadas com conjuntos de campos idênticos a serem tratadas como tabelas internas à parte, quando do contrário seriam concatenadas automaticamente.

Sintaxe:

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

Por padrão, se for carregada uma tabela que contenha um número idêntico de campos e nomes de campo correspondentes a uma tabela carregada anteriormente no script, o Qlik Sense concatenará automaticamente essas duas tabelas. Isso acontecerá mesmo que a segunda tabela tenha um nome diferente.

No entanto, se o prefixo do script `NoConcatenate` for incluído antes da instrução de carregamento ou da instrução de seleção da segunda tabela, essas duas tabelas serão carregadas separadamente.

Um caso de uso típico de `NoConcatenate` é quando você pode precisar criar uma cópia temporária de uma tabela para realizar algumas transformações temporárias nessa cópia, mantendo uma cópia dos dados originais. `NoConcatenate` garante que você possa fazer essa cópia sem adicioná-la implicitamente de volta à tabela de origem.

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplo de função

Exemplo	Resultado
`Source: LOAD A,B from file1.csv; CopyOfSource: NoConcatenate LOAD A,B resident Source;`	Uma tabela com `A` e `B` como medidas é carregada. Uma segunda tabela com os mesmos campos é carregada separadamente usando a variável `NoConcatenate`.

Exemplo 1: Concatenação implícita

Script de carregamento e resultados

Visão geral

Neste exemplo, você adicionará dois scripts de carregamento em ordem sequencial.

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados inicial com datas e valores que é enviado para uma tabela denominada `Transactions`.

Primeiro script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

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

Primeira tabela de resultados

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

id	date	amount
6	09/22/2018	59.18
7	09/23/2018	177.42

Segundo script de carregamento

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um segundo conjunto de dados com campos idênticos é enviado para uma tabela denominada `Sales`.

```
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
];
```

Resultados

Carregue os dados e vá até a tabela.

Segunda tabela de resultados

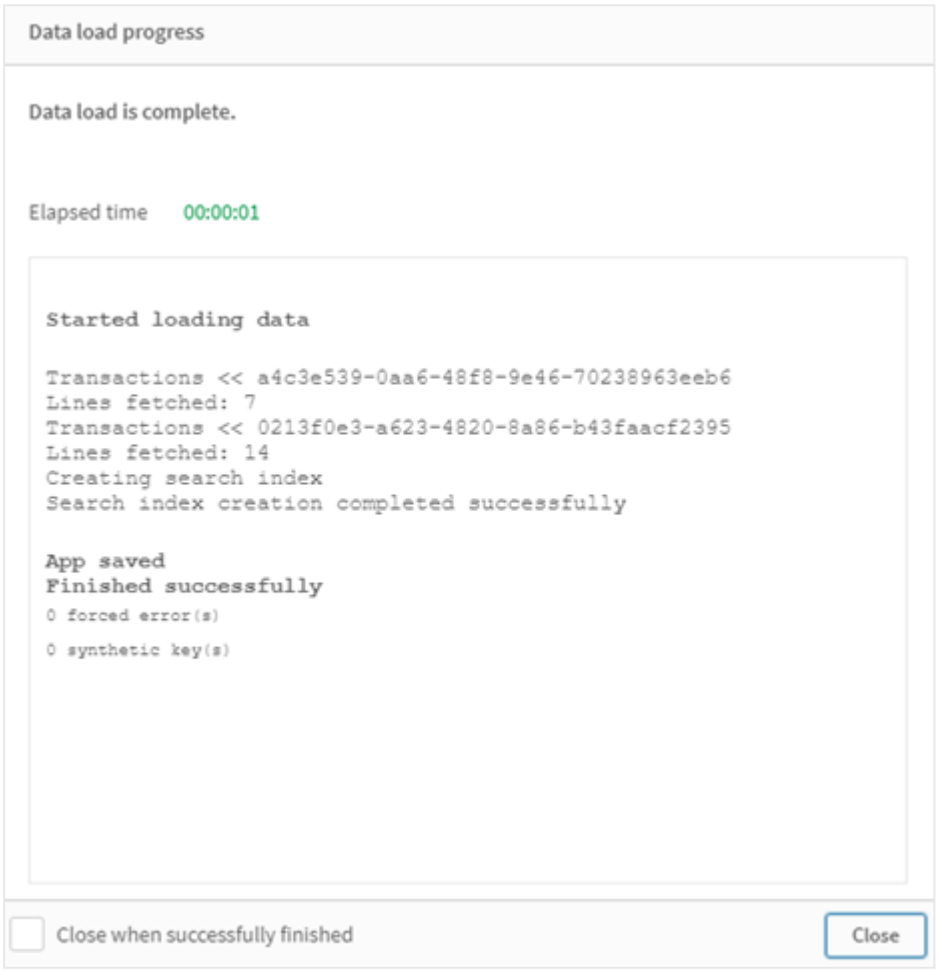
id	date	amount
1	08/30/2018	23.56
2	09/07/2018	556.31
3	09/16/2018	5.75
4	09/22/2018	125.00
5	09/22/2018	484.21
6	09/22/2018	59.18
7	09/23/2018	177.42
8	10/01/2018	164.27
9	10/03/2018	384.00
10	10/06/2018	25.82
11	10/09/2018	312.00

id	date	amount
12	10/15/2018	4.56
13	10/16/2018	90.24
14	10/18/2018	19.32

Quando o script é executado, a tabela `Sales` é implicitamente concatenada na tabela `Transactions` existente pelo fato de os dois conjuntos de dados compartilharem um número idêntico de campos, com nomes de campo idênticos. Isso acontece apesar de a segunda tag de nome da tabela tentar nomear o conjunto de resultados `'Sales'`.

Examinando o registro de **Progresso do carregamento de dados**, é possível ver que o conjunto de dados de vendas está implicitamente concatenado.

Registro de progresso do carregamento de dados mostrando dados de Transactions sendo concatenados implicitamente.



Exemplo 2: Cenário de caso de uso

Script de carregamento e resultados

Visão geral

Nesse cenário de caso de uso, você tem:

- Um conjunto de dados de transações com:
    - id
    - date
    - amount (em GBP)
- Uma tabela de moedas com:
    - Taxas de conversão de USD para GBP
- Um segundo conjunto de dados de transações com:
    - id
    - date
    - amount (em USD)

Você carregará cinco scripts em ordem sequencial.

- O primeiro script de carregamento contém um conjunto de dados inicial com datas e valores em GBP que é enviado para uma tabela denominada `Transactions`.
- O segundo script de carregamento contém:
    - Um segundo conjunto de dados com datas e valores em USD que é enviado para uma tabela denominada `Transactions_in_USD`.
    - O prefixo `noconcatenate` que é colocado antes da instrução de carregamento do conjunto de dados `Transactions_in_USD` para evitar a concatenação implícita.
- O terceiro script de carregamento contém o prefixo `join` que será usado para criar uma taxa de câmbio entre GBP e USD na tabela `Transactions_in_USD`.
- O quarto script de carregamento contém o prefixo `concatenate` que adicionará `Transactions_in_USD` à tabela inicial `Transactions`.
- O quinto script de carregamento contém a instrução `drop table` que removerá a tabela `Transactions_in_USD` cujos dados foram concatenados à tabela `Transactions`.

Primeiro script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- id
- date
- amount

Primeiros resultados do script de carregamento

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

A tabela mostra o conjunto de dados inicial com valores em GBP.

Segundo script de carregamento

```
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
];
```

Resultados

Carregue os dados e vá até a tabela.

Resultados do segundo script de carregamento

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
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

Você verá que o segundo conjunto de dados da tabela `Transactions_in_USD` foi adicionado.

Terceiro script de carregamento

Esse script de carregamento une uma taxa de câmbio de USD para GBP à tabela `Transactions_in_USD`.

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

Resultados

Carregue os dados e acesse o Visualizador do modelo de dados. Selecione a tabela `Transactions_in_USD`, e você verá que cada registro existente tem um valor de campo de “rate” de 0,7.

Quarto script de carregamento

Usando carregamento residente, esse script de carregamento concatenará a tabela `Transactions_in_USD` com a tabela `Transactions` depois de converter os valores em USD.

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

Resultados

Carregue os dados e vá até a tabela. Você verá novas entradas com valores em GBP das linhas de oito a quatorze.

Resultados do quarto script de carregamento

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	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

Quinto script de carregamento

Esse script de carregamento eliminará as entradas duplicadas da tabela de resultados do quarto script de carregamento, deixando apenas entradas com valores em GBP.

```
drop tables Transactions_in_USD;
```

Resultados

Carregue os dados e vá até a tabela.

Resultados do quinto script de carregamento

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	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

Após o carregamento do quinto script de carregamento, a tabela de resultados mostra todas as quatorze transações que existiam em ambos os conjuntos de dados de transações. No entanto, as transações de 8 a 14 tiveram seus valores convertidos em GBP.

Se removermos o prefixo `NoConcatenate` usado antes de `Transactions_in_USD` no segundo script de carregamento, o script falhará com o erro: “Tabela `Transactions_in_USD` não encontrada”. Isso ocorre porque a tabela `Transactions_in_USD` teria sido concatenada automaticamente com a tabela `Transactions` original.

Only

A palavra chave de script é usada como uma função de agregação ou como parte da sintaxe em prefixos de recarga parcial e .**OnlyAddReplaceMerge**

Outer

O prefixo **Join** explícito pode ser precedido pelo prefixo **Outer** para especificar uma junção externa. Em uma junção externa, todas as combinações entre as duas tabelas são geradas. A tabela resultante conterá

combinações de valores de campo das tabelas de dados brutas em que os valores de campo de ligação são representados em uma ou ambas as tabelas. A palavra-chave **Outer** é opcional e é o tipo de junção padrão usado quando um prefixo join não é especificado.

Sintaxe:

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

Argumentos:

Argumentos

Argumento	Descrição
tablename	A tabela nomeada a ser comparada com a tabela carregada.
loadstatementou selectstatement	O comando **LOAD** ou **SELECT** da tabela carregada.

Exemplo

Script de carregamento

Adicione o script de exemplo ao seu aplicativo e execute-o. Para ver o resultado, adicione os campos listados na coluna de resultados a uma pasta no seu aplicativo.

```
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 ];
```

Tabela resultante

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

Explicação

Neste exemplo, as duas tabelas, Table1 e Table2, são mescladas em uma única tabela rotulada como Table1. Em casos como este, o prefixo **outer** é frequentemente usado para unir várias tabelas em uma única tabela para realizar agregações sobre os valores de uma única tabela.

Carregamento parcial

Um carregamento total sempre começa excluindo todas as tabelas no modelo de dados existente e, em seguida, executa o script de carregamento.

Um carregamento parcial não fará isso. Em vez disso, ele mantém todas as tabelas no modelo de dados e depois executa apenas os comandos **Load** e **Select** precedidos por um prefixo **Add**, **Merge** ou **Replace**. Outras

tabelas de dados não são afetadas pelo comando. O argumento **only** indica que o comando deve ser executado apenas durante carregamentos parciais e deve ser ignorado durante carregamentos totais. A tabela a seguir resume a execução da instrução para carregamentos parciais e totais.

Instrução	**Carregamento total**	**Carregamento parcial**
Load ...	A instrução será executada	A instrução não será executada
Adicionar/substituir/mesclar carregamento...	A instrução será executada	A instrução será executada
Adicionar/substituir/mesclar somente carregamento...	A instrução não será executada	A instrução será executada

Carregamentos parciais têm vários benefícios em comparação com carregamentos completos:

- Mais rápidos, porque somente os dados recém-alterados precisam ser carregados. Com grandes conjuntos de dados, a diferença é significativa.
- Menos memória é consumida, porque menos dados são carregados.
- Mais confiáveis, porque as consultas aos dados de origem são executadas mais rapidamente, reduzindo o risco de problemas de rede.

Para que o carregamento parcial funcione corretamente, o aplicativo deve ser aberto com dados antes que um carregamento parcial seja disparado.

Execute um carregamento parcial usando o botão **Carregar**. Você também pode usar o Qlik Engine JSON API.

Limitações

Uma recarga parcial falhará se houver comandos com referências a tabelas que existiam durante a recarga completa, mas não durante a recarga parcial.

Exemplo

Comandos de exemplo

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

Onde <Table_removed_after_full_reload> é uma tabela que existia na recarga completa, mas não na recarga parcial.

Solução alternativa

Como solução alternativa, você pode cercar o comando com a seguinte instrução if:

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

Um carregamento parcial pode remover valores dos dados. No entanto, isso não será refletido na lista de valores distintos, que é uma tabela mantida internamente. Assim, após um carregamento parcial, a lista conterá todos os valores distintos que existiam no campo desde o último carregamento parcial, o que pode ser superior ao que existe atualmente após esse carregamento parcial. Isso afeta a saída das funções FieldValueCount() e FieldValue(). FieldValueCount () poderia retornar um número maior que o número atual de valores de campos.

Exemplo

Exemplo 1

Script de carregamento

Adicione o script de exemplo ao seu aplicativo e faça um carregamento parcial. Para ver o resultado, adicione os campos listados na coluna de resultados a uma pasta no seu aplicativo.

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

Resultado

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

Explicação

A instrução só é executada durante um carregamento parcial. Se o prefixo “distinct” for omitido, a contagem do campo **Num** aumentará a cada carregamento parcial subsequente.

Exemplo 2

Script de carregamento

Adicione o script de exemplo ao seu aplicativo. Faça um carregamento total e veja o resultado. Em seguida, faça um carregamento parcial e veja o resultado. Para ver os resultados, adicione os campos listados na coluna de resultados a uma pasta no seu aplicativo.

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

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

Resultado

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

Explicação

A primeira tabela é carregada durante um carregamento completo e a segunda tabela simplesmente substitui a primeira tabela durante um carregamento parcial.

Replace

A palavra chave de script **Replace** é usada como uma função de string ou como um prefixo em uma recarga parcial.

Replace

O prefixo **Replace** pode ser adicionado a qualquer comando **LOAD** ou **SELECT** no script para especificar que a tabela carregada deve substituir outra tabela. Ele também especifica que esse comando deve ser executado em um carregamento parcial. O prefixo **Replace** também pode ser usado em um comando **Map**.

Para que o carregamento parcial funcione corretamente, o aplicativo deve ser aberto com dados antes que um carregamento parcial seja disparado.

Execute um carregamento parcial usando o botão **Carregar**. Você também pode usar o Qlik Engine JSON API.

Sintaxe:

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

```
Replace [only] mapstatement
```

Durante um carregamento normal (não parcial), a construção **Replace LOAD** funcionará como um comando **LOAD** normal, mas precedida por **Drop Table**. Em primeiro lugar, a tabela antiga será descartada e depois registros serão gerados e armazenados como uma nova tabela.

Se o prefixo **Concatenate** for usado, ou se existir uma tabela com o mesmo conjunto de campos, ela será a tabela relevante a ser descartada. De outra forma, não haverá uma tabela para descartar, e a construção **Replace LOAD** será idêntica a um **LOAD** normal.

Um carregamento parcial fará o mesmo. A única diferença é que há sempre uma tabela da execução do script anterior para descartar. A construção **Replace LOAD** sempre descartará primeiro a tabela antiga e, em seguida, criará uma nova.

A instrução **Replace Map...Using** faz com que o mapeamento ocorra também durante a execução parcial do script.

Argumentos:

Argumentos

Argumento	Descrição
only	Um qualificador opcional indicando que o comando deve ser executado somente durante carregamentos parciais. Ele deve ser desconsiderado durante carregamentos normais (não parciais).

Exemplos e resultados:

Exemplo	Resultado
`Tab1: Replace LOAD * from File1.csv;`	Durante as recargas normais e parciais, a tabela Tab1 do Qlik Sense é inicialmente descartada. Em seguida, novos dados são carregados de File1.csv e armazenados em Tab1.
`Tab1: Replace only LOAD * from File1.csv;`	Durante a recarga normal, esse comando é desconsiderado. Durante a recarga parcial, qualquer tabela do Qlik Sense anteriormente denominada Tab1 é inicialmente descartada. Em seguida, novos dados são carregados de File1.csv e armazenados em Tab1.
`Tab1: LOAD a,b,c from File1.csv; Replace LOAD a,b,c from File2.csv;`	Durante a recarga normal, primeiro o arquivo File1.csv é lido na tabela Tab1 do Qlik Sense, mas, em seguida, é imediatamente descartado e substituído por novos dados carregados de File2.csv. Todos os dados de File1.csv são perdidos. Durante o carregamento parcial, toda a tabela Tab1 do Qlik Sense é inicialmente descartada. Em seguida, é substituída por novos dados carregados de File2.csv.
`Tab1: LOAD a,b,c from File1.csv; Replace only LOAD a,b,c from File2.csv;`	Durante uma recarga normal, os dados são carregados de File1.csv e armazenados na tabela Tab1 do Qlik Sense. File2.csv é desconsiderado. Durante a recarga parcial, toda a tabela Tab1 do Qlik Sense é inicialmente descartada. Em seguida, é substituída por novos dados carregados de File2.csv. Todos os dados de File1.csv são perdidos.

Right

Os prefixos **Join** e **Keep** podem ser precedidos pelo prefixo **right**.

Se usado antes de **join**, especifica que right join deve ser usado. A tabela resultante só conterá combinações de valores de campo das tabelas de dados brutas nas quais os valores de campo de link são representados na segunda tabela. Se usado antes de **keep**, especificará que a primeira tabela de dados brutos deverá ser reduzida à sua interseção comum com a segunda tabela antes de ser armazenada no Qlik Sense.

Você estava procurando pela função de caracteres pelo mesmo nome? Consulte: Right (page 1474)

Sintaxe:

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

Argumentos:

Argumentos

Argumento	Descrição
tablename	A tabela nomeada a ser comparada com a tabela carregada.
loadstatementou selectstatement	O comando **LOAD** ou **SELECT** da tabela carregada.

Exemplo

Script de carregamento

Adicione o script de exemplo ao seu aplicativo e execute-o. Para ver o resultado, adicione os campos listados na coluna de resultados a uma pasta no seu aplicativo.

```
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 ];
```

Resultado

Tabela resultante

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

Explicação

Este exemplo demonstra a saída de Junção direita, em que somente os valores presentes na segunda tabela (direita) são unidos.

Sample

O prefixo **sample** em um comando **LOAD** ou **SELECT** é usado para carregar uma amostra aleatória de registros de uma fonte de dados.

Sintaxe:

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

A expressão avaliada não define a porcentagem de registros do conjunto de dados que serão carregados no aplicativo Qlik Sense, mas a probabilidade de cada registro lido ser carregado no aplicativo. Em outras palavras, especificar um valor `p = 0.5` não significa que 50% do número total de registros serão carregados, mas sim que, para cada registro, haverá 50% de chance de ele ser carregado no aplicativo Qlik Sense.

Argumentos

Argumento	Descrição
p	Uma expressão arbitrária avaliada como um número maior que 0 e menor que ou igual a 1. O número indica a probabilidade de um determinado registro ser lido. Todos os registros serão lidos, mas apenas alguns deles serão carregados no Qlik Sense.

Quando usar

A amostra é útil quando você deseja obter amostras de dados provenientes de uma tabela grande, para entender a natureza dos dados, da distribuição ou do conteúdo do campo. Como ela traz um subconjunto de dados, os carregamentos de dados são mais rápidos, permitindo testes mais rápidos de scripts. Diferentemente de `First`, a função `Sample` traz dados de toda a tabela, em vez de se limitar às primeiras linhas. Isso pode fornecer uma representação mais precisa dos dados em alguns casos.

Os exemplos a seguir mostram dois usos possíveis do prefixo do script `Sample`:

```
Sample 0.15 SQL SELECT * from Longtable;

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

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplo 1: Amostra de uma tabela inline

Script de carregamento e resultados

Visão geral

Neste exemplo, o script carrega um exemplo de conjunto de dados de um conjunto de dados contendo sete registros em uma tabela denominada `Transactions` de uma tabela inline.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `id`
- `amount`

Adicione a seguinte medida:

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

Tabela de resultados

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

Na iteração do carregamento usado neste exemplo, todos os sete registros foram lidos, mas somente quatro registros foram carregados na tabela de dados. Qualquer carregamento de nova execução pode resultar no carregamento de um número diferente e de um conjunto diferente de registros no aplicativo.

Exemplo 2: Amostra de uma tabela gerada automaticamente

Script de carregamento e resultados

Visão geral

Neste exemplo, usando `Autogenerate`, um conjunto de dados de 100 registros é criado com os campos `date`, `id` e `amount`. No entanto, o prefixo `Sample` é usado com um valor de 0,1.

Script de carregamento

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

Autogenerate(100);
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- id
- amount

Adicione a seguinte medida:

Tabela de resultados

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

Na iteração do carregamento usado neste exemplo, sete registros foram carregados do conjunto de dados criado. Mais uma vez, qualquer carregamento de nova execução pode resultar no carregamento de um número diferente e de um conjunto diferente de registros no aplicativo.

Semantic

O prefixo de carregamento `semantic` cria um tipo especial de campo que pode ser usado no Qlik Sensepara conectar e gerenciar dados relacionais, como estruturas em árvore, dados estruturados pai/filho de autorreferência e/ou dados que podem ser descritos como um gráfico.

Observe que o carregamento de `semantic` pode funcionar de forma semelhante aos prefixos *Hierarchy (page 65)* e *HierarchyBelongsTo (page 67)*. Todos os três prefixos podem ser usados como blocos de construção em soluções front-end eficazes para a travessia de dados relacionais.

Sintaxe:

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

Um carregamento semântico espera uma entrada com exatamente três ou quatro campos de largura, com uma definição estrita do que cada campo ordenado representa, como mostra a tabela abaixo:

Campos de carregamento semântico

Nome do campo	Descrição do campo
1º campo:	Essa tag é uma representação do primeiro dos dois objetos entre os quais existe um relacionamento.
2º campo:	Essa tag será usada para descrever a relação "direta" entre o primeiro e o segundo objetos. Se o primeiro objeto for um filho e o segundo objeto for um pai, você poderá criar uma guia de

Nome do campo	Descrição do campo
	relacionamento que indique "pai" ou "pai de" como se estivesse seguindo o relacionamento do filho para o pai.
3º campo:	Essa tag é uma representação do segundo dos dois objetos entre os quais existe um relacionamento.
4º campo:	Esse campo é opcional. Essa tag descreve o relacionamento "regressivo" ou "inverso" entre o primeiro e o segundo objetos. Se o primeiro objeto for filho e o segundo objeto for pai, uma guia de relacionamento poderá indicar "filho" ou "filho de", como se você estivesse seguindo o relacionamento do pai para o filho. Se você não adicionar um quarto campo, a segunda tag de campo será usada para descrever o relacionamento em qualquer direção. Nesse caso, um símbolo de seta é adicionado automaticamente como parte da tag.

O código a seguir é um exemplo do prefixo `semantic`.

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

É permitido e prática típica rotular o terceiro campo da mesma forma que o primeiro campo. Isso cria uma pesquisa de autorreferência, para que você possa seguir os objetos até os objetos relacionados, um passo de cada vez. Se o terceiro campo não tiver o mesmo nome, o resultado final será uma pesquisa simples de um ou mais objetos para seus vizinhos relacionais diretos a apenas um passo de distância, o que é uma saída de pouco uso prático.

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Funções relacionadas

Funções	Interação
Hierarchy (page 65)	O prefixo de carregamento Hierarchy é usado para dividir e organizar nós em estruturas de dados pai/filho e outras estruturas de dados semelhantes a gráficos e transformá-los em tabelas.
HierarchyBelongsTo (page 67)	O prefixo de carregamento HierarchyBelongsTo é usado para localizar e organizar os ancentrais de estruturas de dados pai/filho e outras estruturas de dados semelhantes a gráficos e transformá-los em tabelas.

Exemplo: Criação de um campo especial para conectar relacionamentos usando o prefixo semântico

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados que representa registros de relação geográfica e que é carregado em uma tabela denominada `GeographyTree`.
    - Cada entrada tem um ID no início da linha e um parentID no final da linha.
- O prefixo `semantic`, que adicionará um campo de comportamento especial chamado `Relation`.

Script de carregamento

```
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;
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões.

- `Id`
- `Geography`

Em seguida, crie um painel de filtro com `Relation` como dimensão. Clique em **Edição concluída**.

Tabela de resultados

ID	**Geografia**
1	Mundo
2	Europa
3	Ásia
4	América do Norte
5	América do Sul
6	Reino Unido
7	Alemanha
8	Sweden
9	Coreia do Sul
10	Coreia do Norte
11	China
12	Londres
13	Birmingham

Painel de filtro

Relação
Filho
Pai

Clique em **Europa** na dimensão `Geography` da tabela e clique em **Filho** na dimensão `Relation` no painel de filtro. Observe o resultado esperado na tabela:

Tabela de resultados mostrando "filhos" de Europa

ID	Geografia
6	Reino Unido
7	Alemanha
8	Sweden

Clicar em **Filho** novamente mostrará lugares que são "filhos" do Reino Unido, um passo mais abaixo.

Tabela de resultados mostrando "filhos" do Reino Unido

ID	Geografia
12	Londres
13	Birmingham

Unless

O prefixo e o sufixo **unless** são utilizados para criar uma cláusula condicional que determina se um comando ou uma cláusula exit deve ser avaliada. Pode ser considerado como uma alternativa compacta do comando **if..end if**.

Sintaxe:

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

O **statement** ou o **exitstatement** será executado apenas se **condition** for avaliado comoFalse.

O prefixo **unless** pode ser utilizado em comandos que já possuem um ou vários outros comandos, inclusive prefixos adicionais **unless** ou **when**.

Argumentos

Argumento	Descrição
condition	Uma expressão lógica de avaliação como True ou False.
statement	Qualquer comando de script do Qlik Sense, exceto os de controle.
exitstatement	Uma cláusula **exit for**, **exit do** ou **exit sub** ou um comando **exit script**.

Quando usar

A instrução `Unless` retorna um resultado booleano. Normalmente, esse tipo de função será usado como uma condição quando o usuário quiser carregar ou excluir condicionalmente partes do script.

As linhas a seguir mostram três exemplos de como a função `Unless` pode ser usada:

```
exit script unless A=1;
```

```
unless A=1 LOAD * from myfile.csv;

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

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplo 1: prefixo “unless”

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- A criação da variável A, à qual é especificado um valor de 1.
- Um conjunto de dados que é carregado em uma tabela denominada Transactions, a menos que a variável A = 2.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

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

Tabela de resultados

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

Como a variável A recebe o valor de 1 no início do script, a condição após o prefixo `Unless` é avaliada, retornando um resultado de `FALSE`. Como resultado, o script continua executando a instrução `Load`. Na tabela de resultados, todos os registros da tabela `Transactions` podem ser vistos.

Se o valor dessa variável for definido como igual a 2, nenhum dado será carregado no modelo de dados.

Exemplo 2: sufixo “unless”

Script de carregamento e resultados

Visão geral

Abra o editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento começa carregando um conjunto de dados inicial em uma tabela denominada `Transactions`. O script é então encerrado, a menos que haja menos de 10 registros na tabela `Transactions`.

Se essa condição não resultar no encerramento do script, um conjunto adicional de transações será concatenado na tabela `Transactions`, e esse processo será repetido.

Script de carregamento

```
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 ;
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- id
- date
- amount

Tabela de resultados

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

Há sete registros em cada um dos três conjuntos de dados do script de carregamento.

O primeiro conjunto de dados (com a transação `id` de 1 a 7) é carregado no aplicativo. A condição `Unless` avalia se há menos de 10 linhas na tabela `Transactions`. Isso é avaliado como `TRUE` e, portanto, o segundo conjunto de dados (com as transações `id` de 8 a 14) é carregado no aplicativo. A segunda condição `Unless` avalia se há menos de 10 registros na tabela `Transactions`. Isso é avaliado como `FALSE` e, portanto, o script é encerrado.

Exemplo 3: vários prefixos “unless”

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

Neste exemplo, um conjunto de dados contendo uma transação é criado como uma tabela denominada `Transactions`. Um loop “for” é então acionado, no qual duas instruções “unless” aninhadas são avaliadas:

1. A menos que haja mais de 100 registros na tabela `Transactions`
2. A menos que o número de registros na tabela `Transactions` seja múltiplo de 6

Se essas condições forem `FALSE`, mais sete registros serão gerados e concatenados na tabela `Transactions` existente. Esse processo é repetido até que uma das duas transações retorne um valor de `TRUE`.

Script de carregamento

```
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
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão:`id`.

Tabela de resultados

id
0
1
2
3
4
5
Mais de 30 linhas

As instruções “unless” aninhadas que ocorrem no loop “for” avaliam o seguinte:

1. Há mais de 100 linhas na tabela `Transactions`?
2. O número total de registros na tabela `Transactions` é múltiplo de 6?

Sempre que ambas as instruções “unless” retornam um valor de FALSE, mais sete registros são gerados e concatenados na tabela `Transactions` existente.

Essas instruções retornam um valor de FALSE cinco vezes, momento em que há um total de 36 linhas de dados na tabela `Transactions`.

Depois disso, a segunda instrução `Unless` retorna um valor de TRUE e, portanto, a instrução de carregamento seguinte não será mais executada.

When

O prefixo e o sufixo **when** são utilizados para criar uma cláusula condicional que determina se um comando ou uma cláusula exit deve ser executada. Pode ser considerado como uma alternativa compacta do comando **if..end if**.

Sintaxe:

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

Tipo de dados de retorno: Booleano

No Qlik Sense, o valor booleano “true” é representado por -1, e o valor falso é representado por 0.

A **instrução** ou a **instrução de saída** apenas será executada se a condição for avaliada como TRUE.

O prefixo `Unless` pode ser utilizado em comandos que já possuem um ou vários outros comandos, inclusive prefixos adicionais `when` ou `When`.

Quando usar

A instrução `When` retorna um resultado booleano. Normalmente, esse tipo de função será usado como uma condição quando o usuário quiser carregar ou excluir partes de um script.

Argumentos

Argumento	Descrição
condition	Uma expressão lógica que é avaliada como TRUE ou FALSE
statement	Qualquer comando de script do Qlik Sense, exceto os de controle.
exitstatement	Uma cláusula **exit for**, **exit do** ou **exit sub** ou um comando **exit script**.

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplos de funções

Exemplo	Resultado
`exit script when A=1;`	Quando a instrução `A=1` for avaliada como TRUE, o script será interrompido.
`when A=1 LOAD * from myfile.csv;`	Quando a instrução `A=1` for avaliada como TRUE, o `myfile.csv` será carregado.
`when A=1 unless B=2 drop table Tab1;`	Quando a instrução `A=1` for avaliada como TRUE e `B=2` for avaliada como FALSE, a tabela `Tab1` será descartada.

Exemplo 1: prefixo “when”

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados com datas e valores que é enviado para uma tabela denominada "`Transactions`".
- A instrução `Let` que afirma que a variável `A` foi criada e tem o valor de `1`.
- A condição `When` que fornece a condição de que, se `A` for igual `1`, o script continuará sendo carregado.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- id
- date
- amount

Tabela de resultados

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

Como a variável A recebe o valor de 1 no início do script, a condição após o prefixo When é avaliada e retorna o resultado TRUE. Como ela retorna um resultado TRUE, o script continua executando a instrução de carregamento. Todos os registros da tabela de resultados podem ser vistos.

Se esse valor de variável fosse definido como qualquer valor diferente de 1, nenhum dado seria carregado no modelo de dados.

Exemplo 2: sufixo “when”

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Três conjuntos de dados com datas e valores enviados para uma tabela denominada “Transactions”.
  - O primeiro conjunto de dados contém transações de 1 a 7.
  - O segundo conjunto de dados contém transações de 8 a 14.
  - O terceiro conjunto de dados contém transações de 15 a 21.
- Uma condição When que determina se a tabela “Transactions” contém mais de dez linhas. Se alguma das instruções When for avaliada como TRUE, o script de carregamento será interrompido. Essa condição é colocada no final de cada um dos três conjuntos de dados.

Script de carregamento

```
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 ;
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- id
- date
- amount

Tabela de resultados

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

Há sete transações em cada um dos três conjuntos de dados. O primeiro conjunto de dados contém a transação de 1 a 7 e é carregado no aplicativo. A condição `When` após essa instrução de carregamento é avaliada como FALSE, porque há menos de dez linhas na tabela "`Transactions`". O script de carregamento continua até o próximo conjunto de dados.

O segundo conjunto de dados contém a transação de 8 a 14 e é carregado no aplicativo. A segunda condição `When` é avaliada como TRUE porque há mais de dez linhas na tabela "`Transactions`". Portanto, o script é encerrado.

Exemplo 3: vários prefixos "when"

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo uma única transação é criado como uma tabela denominada "`Transactions`".
- Um loop `For` que é acionado contém duas condições `When` aninhadas que avaliam se:

1. Há menos de 100 registros na tabela "Transactions".
2. O número de registros na tabela "Transactions" não é múltiplo de 6.

Script de carregamento

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

For i = 1 to 100
       when NoOfRows('Transactions') < 100 when mod(NoOfRows('Transactions'),6) <> 0
               Concatenate
    Load
       if(isnull(Peek(id)),1,peek(id)+1) as id
      Autogenerate 7;
next i
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão:

- id

A tabela de resultados mostra apenas os primeiros cinco IDs de transação, mas o script de carregamento cria 36 linhas e depois termina quando a condição When é atendida.

Tabela de resultados

id
0
1
2
3
4
5
Mais de 30 linhas

As condições When aninhadas no loop For avaliam as seguintes perguntas:

- Há menos de 100 linhas na tabela "Transactions"'?
- O número total de registros na tabela "Transactions" não é múltiplo de seis?

Sempre que ambas as condições When retornam um valor TRUE, mais sete registros são gerados e concatenados na tabela "Transactions" existente.

As condições `When` retornam um valor TRUE cinco vezes. Nesse ponto, há um total de 36 linhas de dados na tabela "`Transactions`".

Quando 36 linhas de dados são criadas na tabela "`Transactions`", a segunda instrução `When` retorna um valor FALSE e, portanto, a instrução de carregamento seguinte não será mais executada.

2.5 Comandos regulares de script

Comandos comuns geralmente são usados para manipular dados de uma forma ou de outra. Esses comandos podem ser escritos em qualquer quantidade de linhas no script e devem sempre ser encerrados por um ponto-e-vírgula ";".

Todas as palavras-chave do script podem ser digitadas com qualquer combinação de caracteres maiúsculos e minúsculos. No entanto, os nomes de campos e de variáveis usados nos comandos diferenciam maiúsculas de minúsculas.

Visão geral dos comandos regulares de script

Cada função é descrita adicionalmente após a visão geral. Você também pode clicar no nome da função na sintaxe para acessar imediatamente os detalhes dessa função específica.

Alias
A instrução **alias** é usada para definir um alias de acordo com o qual um campo será renomeado sempre que ele ocorrer no script que segue.

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

Autonumber
Esse comando cria um valor inteiro exclusivo para cada valor distinto avaliado em um campo encontrado durante a execução do script.

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

Binary
O comando **binary** é usado para carregamento dos dados de outro aplicativo QlikView ou documento do , inclusive dados de acesso da seção.

```
Binary  [path] filename
```

comment
Fornece um meio de exibir os comentários de campo (metadados) das bases de dados e planilhas. Os nomes de campo não presentes no aplicativo serão ignorados. Se houver várias ocorrências de um nome de campo, será usado o último valor.

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

comment table
Fornece um meio de exibir os comentários de tabela (metadados) das bases de dados ou planilhas.

```
Comment table tablelist using mapname
Comment table tablename with comment
```

Connect

Esta funcionalidade não está disponível no Qlik Sense SaaS.

O comando **CONNECT** é usado para definir o acesso do Qlik Sense a uma base de dados geral por meio da interface OLE DB/ODBC. Para ODBC, a fonte de dados deve ser inicialmente especificada usando o administrador ODBC.

```
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

O comando **Declare** é usado para criar definições de campo, em que você pode definir as relações entre campos ou funções. Um conjunto de definições de campo pode ser usado para gerar campos derivados automaticamente, o que pode ser usado como dimensões. Por exemplo, é possível criar uma definição de calendário e usá-la para gerar dimensões relacionadas, como ano, mês, semana e dia, a partir de um campo de data.

```
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

O comando **Derive** é usado para gerar campos derivados com base em uma definição de campo criada com um comando **Declare**. Você pode especificar a partir de quais campos de dados derivar os campos ou derivá-los explicitamente ou implicitamente com base em tags de campos.

```
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

O comando **DIRECT QUERY** permite acessar tabelas por meio de uma conexãoODBC ou OLE DB usando a função do Direct Discovery.

```
Direct Query [path]
```

Directory

O comando **Directory** define o diretório para procurar os arquivos de dados em comandos **LOAD** subsequentes, até que um novo comando **Directory** seja feito.

```
Directory [path]
```

Disconnect

O comando **Disconnect** termina a conexão ODBC/OLE DB/Personalizada atual. Esse comando é opcional.

```
Disconnect
```

drop field

É possível descartar um ou vários campos do Qlik Sense do modelo de dados, e consequentemente da memória, a qualquer momento durante a execução do script usando um comando **drop field**. A propriedade "distinct" de uma tabela é removida após uma declaração **drop field**.

*Tanto **drop field** quanto **drop fields** são formas permitidas e não apresentam diferença no efeito. Se nenhuma tabela for especificada, o campo será descartado de todas as tabelas onde ele ocorre.*

```
Drop field  fieldname [ , fieldname2 ...] [from tablename1 [ , tablename2
...]]
drop fields fieldname [ , fieldname2 ...] [from tablename1 [ , tablename2
...]]
```

drop table

É possível descartar uma ou várias tabelas internas do Qlik Sense do modelo de dados, e consequentemente da memória, a qualquer momento durante a execução do script usando um comando **drop table**.

*As formas **drop table** e **drop tables** são aceitas.*

```
Drop table  tablename [, tablename2 ...]
drop tables[ tablename [, tablename2 ...]
```

Execute

O comando **Execute** é utilizado para executar outros programas durante o carregamento de dados do Qlik Sense. Por exemplo, para fazer conversões que sejam necessárias.

```
Execute commandline
```

FlushLog

O comando **FlushLog** força o Qlik Sense a escrever o conteúdo do buffer do script em um arquivo de log do script.

```
FlushLog
```

Force

O comando **force** força o Qlik Sense a interpretar os nomes de campo e valores de campo de comandos **LOAD** e **SELECT** subsequentes como se estivessem escritos apenas em letras maiúsculas, apenas em letras minúsculas, sempre em maiúsculas ou como são exibidos (em letras maiúsculas e minúsculas). Esse comando permite associar os valores de campo das tabelas criadas de acordo com convenções diferentes.

```
Force ( capitalization | case upper | case lower | case mixed )
```

LOAD

A declaração **LOAD** carrega campos de um arquivo, de dados definidos no script, de uma tabela de entrada carregada anteriormente, de uma página da Web, do resultado de um comando **SELECT** subsequente ou gerando dados automaticamente. Também é possível carregar dados de conexões analíticas.

```
 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

O comando **let** é um complemento ao comando **set**, usado para definir variáveis de script. O comando **let**, ao contrário do comando **set**, avalia a expressão no lado direito do sinal de igual "=" no tempo de execução do script antes de ser atribuída à variável.

```
Let variablename=expression
```

Loosen Table

Uma ou mais tabelas de dados internas do Qlik Sense podem ser declaradas explicitamente como parcialmente desconectadas durante a execução do script usando um comando **Loosen Table**. Quando uma tabela é parcialmente desconectada, todas as associações entre os valores de campo da tabela são removidos. Um efeito semelhante pode ser conseguido por meio do carregamento de cada campo da tabela parcialmente desconectada como tabelas independentes desconectadas. A tabela parcialmente desconectada pode ser útil durante os testes para isolar temporariamente diferentes partes da estrutura de dados. Uma tabela parcialmente desconectada pode ser identificada pelas linhas pontilhadas no visualizador de tabelas. O uso de um ou mais comandos **Loosen Table** no script fará com que o Qlik Sense desconsidere qualquer configuração das tabelas como parcialmente desconectadas feita antes da execução do script.

```
tablename [ , tablename2 ...]
Loosen Tables tablename [ , tablename2 ...]
```

Map ... using

O comando **map ... using** é usado para mapear um determinado valor de campo ou uma expressão para os valores de uma tabela de mapeamento específica. A tabela de mapeamento é criada pelo comando **Mapping**.

```
Map  *fieldlist Using  mapname
```

NullAsNull

O comando **NullAsNull** desativa a conversão de valores NULL em strings anteriormente definidas pela declaração **NullAsValue**.

```
NullAsNull *fieldlist
```

NullAsValue

O comando **NullAsValue** especifica para quais campos os NULL encontrados devem ser convertidos em um valor.

```
NullAsValue *fieldlist
```

Qualify

O comando **Qualify** é usado para alterar a qualificação dos nomes de campo, ou seja, nomes de campo receberão o nome da tabela como um prefixo.

```
Qualify *fieldlist
```

Rem

A declaração **rem** é utilizada para inserir comentários no script ou para desativar temporariamente comandos do script sem removê-los.

```
Rem string
```

Rename Field

Essa função de script renomeia um ou mais campos do Qlik Sense existentes depois que eles foram carregados.

```
Rename field (using mapname|oldname to newname{ , oldname to newname })
```

```
Rename Fields (using mapname | oldname to newname{ , oldname to newname })
```

Rename Table

Essa função de script renomeia uma ou mais tabelas internas existentes do Qlik Sense depois que elas foram carregadas.

```
Rename table  (using mapname | oldname to newname{ , oldname to newname })
Rename Tables (using mapname | oldname to newname{ , oldname to newname })
```

Section

Com o comando **section**, é possível definir se os comandos **LOAD** e **SELECT** subsequentes devem ser considerados como dados ou como uma definição dos direitos de acesso.

```
Section (access | application)
```

Select

A seleção de campos de uma fonte de dados ODBC ou provedor OLE DB é feita usando declarações SQL **SELECT** padrão. No entanto, a aceitação dos comandos **SELECT** dependem do driver ODBC ou provedor OLE DB utilizados.

```
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

O comando **set** é usado para definir as variáveis do script. Essas variáveis podem ser utilizadas para substituir strings, caminhos, unidades e assim por diante.

```
Set variablename=string
```

Sleep

O comando **sleep** pausa a execução do script pelo tempo especificado.

```
Sleep n
```

SQL

O comando **SQL** permite enviar um comando SQL arbitrário usando uma conexão ODBC ou OLE DB.

```
SQL sql_command
```

SQLColumns

O comando **sqlcolumns** retorna um conjunto de campos que descreve as colunas de uma fonte de dados ODBC ouOLE DB com a qual é feita uma **connect**.

```
SQLColumns
```

SQLTables

O comando **sqltables** retorna um conjunto de campos que descreve as tabelas de uma fonte de dados ODBC ouOLE DB com a qual é feita uma **connect**.

```
SQLTables
```

SQLTypes

O comando **sqltypes** retorna um conjunto de campos que descreve os tipos de uma fonte de dados ODBC ou OLE DB com a qual é feita uma **connect**.

```
SQLTypes
```

Star

É possível definir os caracteres utilizados para representar o conjunto de todos os valores de um campo na base de dados usando o comando **star**. Isso afeta os comandos **LOAD** e **SELECT** subsequentes.

```
Star is [ string ]
```

Store

A instrução **Store** cria um arquivo QVD, Parquet, CSV ou TXT.

```
Store [ *fieldlist from] table into filename [ format-spec ];
```

Tag

Esse comando de script fornece uma maneira de atribuir tags a um ou mais campos ou tabelas. Se for feita uma tentativa de marcar um campo ou uma tabela não presente no aplicativo, a marcação será ignorada. Se houver ocorrências de um nome de campo ou tag em conflito, o último valor será usado.

```
Tag[field|fields] fieldlist with tagname
Tag [field|fields] fieldlist using mapname
Tag table tablelist with tagname
```

Trace

A declaração **trace** grava a string na janela **Progresso da execução do script** e no arquivo de log do script, quando utilizada. É muito útil para finalidades de depuração. Usando expansões $ de variáveis calculadas antes da declaração **trace**, você pode personalizar a mensagem.

```
Trace string
```

Unmap

O comando **Unmap** desativa o mapeamento do valor de campo especificado por um comando **Map ... Using** prévio de campos subsequentemente carregados.

```
Unmap *fieldlist
```

Unqualify

O comando **Unqualify** é utilizado para desativar a qualificação de nomes de campos previamente ativada pelo comando **Qualify**.

```
Unqualify *fieldlist
```

Untag

Esse comando de script fornece uma maneira de remover tags de campos ou tabelas. Se for feita uma tentativa de desmarcar um campo ou uma tabela não presente no aplicativo, a desmarcação será ignorada.

```
Untag[field|fields] fieldlist with tagname
Tag [field|fields] fieldlist using mapname
Tag table tablelist with tagname
```

Alias

A instrução **alias** é usada para definir um alias de acordo com o qual um campo será renomeado sempre que ele ocorrer no script que segue.

Sintaxe:

```
alias fieldname as aliasname {,fieldname as aliasname}
```

Argumentos:

Argumentos

Argumento	Descrição
fieldname	O nome do campo nos dados de fonte
aliasname	Um pseudônimo que você deseja usar

Exemplos e resultados:

Exemplo	Resultado
`Alias ID_N as NameID;`	
`Alias A as Name, B as Number, C as Date;`	As alterações de nome definidas com essa declaração são usadas em todas as declarações **SELECT** e **LOAD** subsequentes. É possível definir um novo alias para um nome de campo usando uma nova declaração **alias** em qualquer posição subsequente no script.

AutoNumber

Esse comando cria um valor inteiro exclusivo para cada valor distinto avaliado em um campo encontrado durante a execução do script.

Você também pode usar a função *autonumber (page 584)* dentro de uma instrução **LOAD**, mas isso tem algumas limitações quando você deseja usar uma carga otimizada. Você pode criar uma carga otimizada carregando os dados de um arquivo **QVD** primeiro e depois usando a instrução **AutoNumber** para converter valores em teclas de símbolos.

Sintaxe:

```
AutoNumber *fieldlist [Using namespace] ]
```

Argumentos:

Argumentos

Argumento	Descrição
*lista de campos	Uma lista separada por vírgulas dos campos em que os valores devem ser substituídos por um valor inteiro exclusivo. Você pode usar os caracteres curinga ? e * nos nomes dos campos para incluir todos os campos com nomes correspondentes. Você também pode usar * para incluir todos os campos. Você precisa citar nomes de campo quando curingas forem usados.
espaço para nome	O**Using** `espaço de nomes` é opcional. Você poderá usar essa opção se quiser criar um espaço para nome em que valores idênticos em campos diferentes compartilham a mesma chave. Se você não usar essa opção, todos os campos terão um índice de chave separado.

Limitações:

Quando você tem várias instruções **LOAD** no script, é necessário colocar a instrução **AutoNumber** após a instrução **LOAD** final.

Exemplo - script com AutoNumber

Exemplo de script

Neste exemplo, os dados são carregados primeiro sem o comando **AutoNumber**. O comando **AutoNumber** é então adicionado para mostrar o efeito.

Dados usados no exemplo

Carregue os seguintes dados como um carregamento inline no editor de carregamento de dados para criar o exemplo de script abaixo. Deixe o comando **AutoNumber** assinalado como texto por enquanto.

```
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;
```

Criar visualizações

Crie duas visualizações de tabela em uma Qlik Sense do Qlik Cloud. Adicione **KeyToOtherTable**, **Region**, **Year**, **Month** e **Sales** como dimensões à primeira tabela. Adicione **KeyToOtherTable**, **Region**, **Year**, **Month** e **Budget** como dimensões à segunda tabela.

Resultado

Tabela 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

Tabela 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

Explicação

O exemplo mostra um campo composto **KeyToOtherTable** que vincula as duas tabelas. **AutoNumber** não é usado. Observe o comprimento dos valores de **KeyToOtherTable**.

Adicionar o comando AutoNumber

Remova o comentário do comando **AutoNumber** no script de carregamento.

```
AutoNumber KeyToOtherTable;
```

Resultado

Tabela 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

KeyToOtherTable	Region	Year	Month	Sales
4	South	2013	May	367
4	South	2014	June	645

Tabela 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

Explicação

Os valores de campos de **KeyToOtherTable** foram substituídos por valores inteiros exclusivos e, como resultado, o comprimento dos valores de campos foi reduzido, conservando memória. Os campos-chave em ambas as tabelas são afetados por **AutoNumber**, e as tabelas permanecem vinculadas. O exemplo é breve para fins de demonstração, mas seria significativo com uma tabela contendo um grande número de linhas.

Binary

O comando **binary** é usado para carregamento dos dados de outro aplicativo Qlik Sense ou documento do QlikView, inclusive dados de acesso da seção. Outros elementos do aplicativo não estão incluídos, por exemplo, pastas, histórias, visualizações, itens mestre ou variáveis.

Somente um comando **binary** é permitido no script. O comando **binary** deve ser o primeiro comando do script, antes mesmo dos comandos SET, localizados geralmente no começo do script.

Sintaxe:

```
binary [path] filename
```

Argumentos:

Argumentos

Argumento	Descrição
path	O caminho para o arquivo que deve ser uma referência para uma conexão de dados da pasta. Isso é exigido se o arquivo não estiver localizado no diretório de trabalho do Qlik Sense. **Exemplo: *'lib://Table Files/'*** No modo legado de uso de scripts, os seguintes formatos de caminho também são suportados: • absoluto **Exemplo: *c:\data*** • relativo ao aplicativo que contém essa linha de script. **Exemplo: *data***
filename	O nome do arquivo, incluindo a extensão .qvw ou .qvf.

Limitações:

Você não pode usar **binary** para carregar dados de um aplicativo na mesma implantação do Qlik Sense Enterprise fazendo referência ao ID do aplicativo. Somente é possível carregar a partir de um arquivo *.qvf*.

Exemplos

String	Descrição
`Binary lib://DataFolder/customer.qvw;`	Neste exemplo, o arquivo deve estar localizado na conexão de dados **Pasta**. Pode ser, por exemplo, uma pasta criada pelo administrador no servidor do Qlik Sense. Clique em **Criar nova conexão** no editor de carregamento de dados e depois selecione **Pasta** em **Locais de arquivos**.
`Binary customer.qvf;`	Neste exemplo, o arquivo deve estar localizado no diretório de trabalho do Qlik Sense.
`Binary c:\qv\customer.qvw;`	Este exemplo, que utiliza um caminho de arquivo absoluto, somente funcionará no modo de script herdado.

Comment field

Fornece um meio de exibir os comentários de campo (metadados) das bases de dados e planilhas. Os nomes de campo não presentes no aplicativo serão ignorados. Se houver várias ocorrências de um nome de campo, será usado o último valor.

Sintaxe:

```
comment [fields] *fieldlist using mapname
```

```
comment [field] fieldname with comment
```

A tabela de mapas usada deverá ter duas colunas: a primeira contendo nomes de campo e a segunda contendo comentários.

Argumentos:

Argumentos

Argumento	Descrição
**fieldlist*	Uma lista separada por vírgula dos campos a serem comentados. O uso de * como lista de campos indica todos os campos. Os caracteres curingas * e ? são permitidos nos nomes de campo. Poderá ser necessário colocar os nomes de campos entre aspas quando forem utilizados caracteres curinga.
mapname	O nome de uma tabela de mapeamento lida anteriormente em um comando de mapeamento **LOAD** ou **SELECT**.
fieldname	Nome do campo que deve ser comentado.
comment	O comentário que deve ser incluído no campo.

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

Fornece um meio de exibir os comentários de tabela (metadados) das bases de dados ou planilhas.

Os nomes de tabela não presentes no aplicativo serão ignorados. Se houver várias ocorrências de um nome de tabela, será usado o último valor. A palavra-chave pode ser usada para ler comentários de uma fonte de dados.

Sintaxe:

```
comment [tables] tablelist using mapname
comment [table] tablename with comment
```

Argumentos:

Argumentos

Argumento	Descrição
tablelist	(table{,table})
mapname	O nome de uma tabela de mapeamento lida anteriormente em um comando de mapeamento **LOAD** ou **SELECT**.
tablename	O nome da tabela que deve ser comentada.
comment	O comentário que deve ser incluído na tabela.

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

O comando **CONNECT** é usado para definir o acesso do Qlik Sense a uma base de dados geral por meio da interface OLE DB/ODBC. Para ODBC, a fonte de dados deve ser inicialmente especificada usando o administrador ODBC.

Esta funcionalidade não está disponível no Qlik Sense SaaS.

Este comando suporta apenas as conexões de dados da pasta no modo padrão.

Sintaxe:

```
ODBC CONNECT TO connect-string
OLEDB CONNECT TO connect-string
CUSTOM CONNECT TO connect-string
LIB CONNECT TO connection
```

Argumentos:

Argumentos

Argumento	Descrição

connect-string	`connect-string ::= datasourcename { ; conn-spec-item }`
	A string de conexão é o nome da fonte de dados e uma lista opcional de um ou mais itens de especificação de conexão. Se o nome da fonte de dados contiver espaços em branco ou se algum item de especificação de conexão estiver listado, a string de conexão deverá ser colocada entre aspas.
	
	datasourcename deve ser uma fonte de dados ODBC definida ou uma string que defina um provedor OLE DB.
	
	`conn-spec-item ::=`**`DBQ=`**`database_specifier |`**`DriverID=`**`driver_specifier |`**`UID=`**`userid |`**`PWD=`**`password`
	
	Os itens de especificação de conexão possíveis poderão ser diferentes nos diversos bancos de dados. Para alguns bancos de dados, também são possíveis outros itens além dos mencionados acima. Para OLE DB, alguns dos itens específicos de conexão são obrigatórios e não são opcionais.

connection	O nome de uma conexão de dados armazenada no editor da carga de dados.

Se **ODBC** for colocado antes de **CONNECT**, a interface ODBC será usada; caso contrário, OLE DB será usado.

Usar **LIB CONNECT TO** conecta a um banco de dados usando uma conexão de dados armazenados que foi criada no editor da carga de dados.

Example 1:

```
ODBC CONNECT TO 'Sales
DBQ=C:\Program Files\Access\Samples\Sales.mdb';
```

A fonte de dados definida com esse comando é usada por comandos **Select (SQL)** subsequentes, até que um novo comando **CONNECT** seja feito.

Example 2:

```
LIB CONNECT TO 'DataConnection';
```

Connect32

Esse comando é usado da mesma maneira que o comando **CONNECT**, mas força o sistema de 64 bits a usar um provedor de ODBC/OLE DB de 32 bits. Não se aplica ao custom connect.

Connect64

Essa declaração é usada da mesma maneira que a declaração **CONNECT**, mas força o uso de um provedor de 64 bits. Não se aplica ao custom connect.

Declare

O comando **Declare** é usado para criar definições de campo, em que você pode definir as relações entre campos ou funções. Um conjunto de definições de campo pode ser usado para gerar campos derivados automaticamente, o que pode ser usado como dimensões. Por exemplo, é possível criar uma definição de calendário e usá-la para gerar dimensões relacionadas, como ano, mês, semana e dia, a partir de um campo de data.

Você pode usar **Declare** para definir uma nova definição de campo ou criar uma definição de campo com base em uma definição já existente.

Definindo uma nova definição de campo

Sintaxe:

```
definition_name:

Declare [Field[s]] Definition [Tagged tag_list ]

[Parameters parameter_list ]

Fields field_list
```

Argumentos:

Argumento	Descrição
definition_name	Nome da definição de campo, terminado com vírgula. *Não use autoCalendar como nome para definições de campo, pois esse nome está reservado para modelos de calendário gerados automaticamente.* **Exemplo:** `Calendar:`

Argumento	Descrição
tag_list	Uma lista separada por vírgula de tags a serem aplicadas a campos derivados a partir da definição de campo. A aplicação de tags é opcional, mas se você não aplicar as tags usadas para especificar a ordem de classificação, tais como $date, $numeric ou $text, o campo derivado será classificado por ordem de carregamento como padrão. **Exemplo:** `'$date'Thank you for bringing this to our attention, and apologies for the inconvenience.`
parameter_ list	Uma lista de parâmetros separada por vírgula. Um parâmetro é definido na forma de `name=value` e recebe a atribuição de um valor de início, que pode ser ignorado quando uma definição de campo é reutilizada. Opcional. **Exemplo:** `first_month_of_year = 1`
field_list	Uma lista separada por vírgula de campos a serem gerados quando a definição de campo for usada. Um campo é definido na forma de `<expression> As field_name tagged tag`. Use `$1` para referenciar o campo de dados a partir do qual os campos derivados devem ser gerados. **Exemplo:** `Year($1) As Year tagged ('$numeric')`

Exemplo:

```
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')
;
```

O calendário já está definido e você pode aplicá-lo aos campos de data que tenham sido carregados; nesse caso, OrderDate e ShippingDate, por meio de uma cláusula **Derive**.

Reutilizando uma definição de campo existente

Sintaxe:

```
<definition name>:

Declare [Field][s] Definition

Using <existing_definition>

[With <parameter_assignment> ]
```

Argumentos:

Argumento	Descrição
definition_name	Nome da definição de campo, terminado com vírgula. **Exemplo:** `MyCalendar:`
existing_definition	A definição de campo a ser reutilizada quando se estiver criando a nova definição de campo. A nova definição de campo funcionará da mesma forma que a definição na qual ela se baseia, exceto se você usar parameter_assignment para alterar um valor usado nas expressões de campo. **Exemplo:** `Using Calendar`
parameter_assignment	Uma lista de atribuições de parâmetro separada por vírgula. Uma atribuição de parâmetro é definida na forma de `name=value` e ignora o valor de parâmetro existente na definição de campo de base. Opcional. **Exemplo:** `first_month_of_year = 4`

Exemplo:

Nesse exemplo, reutilizamos a definição de calendário criada no exemplo anterior. Nesse caso, queremos usar um ano fiscal que se inicia em abril. Isso é feito atribuindo o valor 4 ao parâmetro first_month_of_year, que afetará o campo DayNumberOfYear que está definido.

O exemplo presume que você tenha usado os dados de amostra e a definição de campo do exemplo anterior.

```
MyCalendar:
DECLARE FIELD DEFINITION USING Calendar WITH first_month_of_year=4;
```

```
DERIVE FIELDS FROM FIELDS OrderDate,ShippingDate USING MyCalendar;
```

Quando você tiver recarregado o script de dados, os campos gerados estarão disponíveis no editor de pasta, com o nomes OrderDate.MyCalendar.* e ShippingDate.MyCalendar.*.

Derive

O comando **Derive** é usado para gerar campos derivados com base em uma definição de campo criada com um comando **Declare**. Você pode especificar a partir de quais campos de dados derivar os campos ou derivá-los explicitamente ou implicitamente com base em tags de campos.

Sintaxe:

```
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
```

Argumentos:

Argumentos

Argumento	Descrição
definition	Nome da definição de campo para usar quando se está derivando campos. **Exemplo:** `Calendar`
field_list	Uma lista de campos de dados separada por vírgula a partir da qual os campos derivados devem ser gerados, com base na definição de campo. Os campos de dados devem ser campos que você já tenha carregado no script. **Exemplo:** `OrderDate, ShippingDate`
tag_list	Uma lista de tags separada por vírgula. Os campos derivados serão gerados para todos os campos de dados com quaisquer tags listadas. A lista de tags deve estar entre colchetes. **Exemplo:** `('$date','$timestamp')`

Exemplos:

- Derivar campos para campos de dados específicos.
  Nesse caso, especificamos os campos OrderDate e ShippingDate.
  `DERIVE FIELDS FROM FIELDS OrderDate,ShippingDate USING Calendar;`
- Derivar campos para todos os campos com uma tag específica.
  Nesse caso, derivamos campos com base em Calendar para todos os campos com uma tag `$date`.
  `DERIVE FIELDS FROM EXPLICIT TAGS ('$date') USING Calendar;`
- Derivar campos para todos os campos com uma tag específica de definição de campo.
  Nesse caso, derivamos campos para todos os campos de dados com a mesma tag que a definição de

campo Calendar que, nesse caso, é $date.
```
DERIVE FIELDS FROM IMPLICIT TAG USING Calendar;
```

Direct Query

O comando **DIRECT QUERY** permite acessar tabelas por meio de uma conexãoODBC ou OLE DB usando a função do Direct Discovery.

Sintaxe:
```
DIRECT QUERY DIMENSION fieldlist [MEASURE fieldlist] [DETAIL fieldlist] FROM tablelist
[WHERE where_clause]
```

As palavras-chaves **DIMENSION**, **MEASURE** e **DETAIL** podem ser usadas em qualquer ordem.

As cláusulas das palavras-chaves **DIMENSION** e **FROM** são obrigatórias em todos os comandos **DIRECT QUERY**. A palavra-chave **FROM** deve aparecer após a palavra-chave **DIMENSION**.

Os campos especificados diretamente após a palavra-chave **DIMENSION** são carregados na memória e podem ser usados para criar associações entre dados da memória e do Direct Discovery.

*O comando **DIRECT QUERY** não pode conter a cláusula **DISTINCT** ou **GROUP BY**.*

Ao usar a palavra-chave **MEASURE**, é possível definir campos que o Qlik Sense tenha conhecimento a um “nível meta”. Os dados reais de um campo de medida residem apenas no banco de dados durante o processo de carga de dados e são recuperados de forma ad hoc, orientados pelas expressões do gráfico que são usadas em uma visualização.

Normalmente, os campos com valores discretos que serão usados como dimensões devem ser carregados com a palavra-chave **DIMENSION**, enquanto os números que serão usados em agregações devem ser selecionados apenas com a palavra-chave **MEASURE**.

DETAIL são campos que fornecem informações ou detalhes, como campos de comentários, que um usuário pode querer exibir em uma tabela. Os campos **DETAIL** não podem ser usados em expressões de gráfico.

Por concepção, o comando **DIRECT QUERY** é neutro de fonte de dados para as fontes de dados que oferecem suporte para SQL. Por essa razão, o mesmo comando **DIRECT QUERY** pode ser usado para diferentes bancos de dados de SQL sem alteração. O Direct Discovery gera consultas apropriadas para o banco de dados, conforme necessário.

A sintaxe nativa da fonte de dados pode ser usada quando o usuário conhece o banco de dados que será consultado e deseja explorar extensões específicas para o banco de dados para o SQL. A sintaxe nativa da fonte de dados é suportada:

- Como expressões de campo nas cláusulas **DIMENSION** e **MEASURE**
- Como o conteúdo da cláusula **WHERE**

Exemplos:

```
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"')
```

Os seguintes termos são usados como palavras-chave e por isso não podem ser usados como nomes de coluna ou de campo sem estar entre aspas: and, as, detach, detail, dimension, distinct, from, in, is, like, measure, native, not, or, where

Argumentos:

Argumento	Descrição
fieldlist	Uma lista de especificações de campo separada por vírgula, *fieldname {, fieldname}*. Uma especificação de campo pode ser um nome de campo, no qual o mesmo nome é usado para o nome de coluna do banco de dados e o nome de campo do Qlik Sense. Ou uma especificação de campo pode ser um "alias de campo", no qual uma expressão do banco de dados ou um nome de coluna recebe um nome de campo do Qlik Sense.
tablelist	Uma lista de nomes de tabelas ou visualizações no banco de dados dos quais os dados serão carregados. Normalmente, serão exibições que contêm um JOIN realizado no banco de dados.

where_clause	A sintaxe completa das cláusulas **WHERE** do banco de dados não está definida aqui, mas a maioria das "expressões relacionais" SQL é permitida, incluindo o uso de chamadas de função, o operador **LIKE** para cadeias de caracteres, **IS NULL** e **IS NOT NULL** e **IN. BETWEEN** não está incluído.
	
	NOT é um operador unário, ao contrário de um modificador em determinadas palavras-chaves.
	
	Exemplos:
	
	`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)`
	O último exemplo não pode ser escrito como:
	
	`WHERE X NOT in (1,2,3)`

Exemplo:

Neste exemplo, é usada uma tabela de banco de dados chamada TableName, contendo os campos Dim1, Dim2, Num1, Num2 e Num3.Dim1 e Dim2 serão carregados no conjunto de dados Qlik Sense.

```
DIRECT QUERY DIMENSTION Dim1, Dim2 MEASURE Num1, Num2, Num3 FROM TableName ;
```

Dim1 e Dim2 estarão disponíveis para uso das dimensões. Num1, Num2 e Num3 estarão disponíveis para agregações. Dim1 e Dim2 também estão disponíveis para agregações. O tipo de agregações para as quais Dim1 e Dim2 podem ser usadas depende de seus tipos de dados. Por exemplo, em muitos casos, os campos **DIMENSION** contêm dados de caracteres, como nomes ou números de conta. Esses campos não podem ser agrupados, mas podem ser contados: `count(Dim1)`.

DIRECT QUERY *são comandos escritos diretamente no editor de script. Para simplificar a construção de instruções* ***DIRECT QUERY****, você pode gerar uma instrução* ***SELECT*** *a partir de uma conexão de dados e, em seguida, editar o script gerado para transformá-lo em uma instrução* ***DIRECT QUERY****. Por exemplo, a instrução* ***SELECT****:*

```
SQL SELECT
 SalesOrderID,
 RevisionNumber,
 OrderDate,
 SubTotal,
 TaxAmt
FROM MyDB.Sales.SalesOrderHeader;
```

pode ser transformada na seguinte instrução ***DIRECT QUERY****:*

```
DIRECT QUERY
 DIMENSION
 SalesOrderID,
 RevisionNumber

MEASURE
 SubTotal,
 TaxAmt

DETAIL
OrderDate

FROM MyDB.Sales.SalesOrderHeader;
```

Listas de campos do Direct Discovery

Uma lista de campo é uma lista das especificações de campo *fieldname {, fieldname}* separada por vírgulas. Uma especificação de campo pode ser um nome de campo, no qual o mesmo nome é usado para o nome de coluna do banco de dados e o nome de campo. Ou uma especificação de campo pode ser um alias de campo, no qual uma expressão do banco de dados ou um nome de coluna recebe um nome de campo do Qlik Sense.

Os nomes de campos podem ser nomes simples ou entre aspas. Um nome simples começa com um caractere Unicode alfabético e é seguido por qualquer combinação de caracteres alfabéticos ou numéricos ou sublinhados. Nomes entre aspas começam com aspas duplas e contêm qualquer sequência de caracteres. Se um nome entre aspas contiver aspas duplas, essas aspas serão representadas com duas aspas adjacentes.

Os nomes de campos do Qlik Sense fazem distinção entre maiúsculas e minúsculas. Os nomes de campo do banco de dados podem ou não fazer distinção entre maiúsculas e minúsculas, dependendo do banco de dados. Uma consulta do Direct Discovery preserva a caixa de todos os identificadores de campo e alias. No exemplo a seguir, o alias "MyState" é usado internamente para armazenar os dados a partir da coluna do banco de dados"STATEID".

```
DIRECT QUERY Dimension STATEID as MyState Measure AMOUNT from SALES_TABLE;
```

Isto difere do resultado de um comando **SQL Select** com um alias. Se o alias não for explicitamente colocado entre aspas, o resultado conterá a caixa padrão da coluna retornada pelo banco de dados de destino. No exemplo a seguir, o comando **SQL Select** para um banco de dados Oracle cria "MYSTATE," com todas as letras maiúsculas, como o alias interno do Qlik Sense, mesmo que o alias seja especificado como letras maiúsculas e minúsculas. O comando **SQL Select** usa o nome da coluna retornado pelo banco de dados, que no caso do Oracle é todo em letras maiúsculas.

```
SQL Select STATEID as MyState, STATENAME from STATE_TABLE;
```

Para evitar este comportamento, use o comando LOAD para especificar o alias.

```
Load STATEID as MyState, STATENAME;
SQL Select STATEID, STATEMENT from STATE_TABLE;
```

Neste exemplo, a coluna "STATEID" é armazenada internamente peloQlik Sense como "MyState".

A maioria das expressões escalares do banco de dados é permitida como especificações de campo. As chamadas de função também podem ser utilizadas nas especificações de campo. As expressões podem conter constantes que sejam boolianas, numéricas ou cadeias de caracteres contidas em aspas simples (aspas simples incorporadas são representadas por aspas simples adjacentes).

Exemplos:

```
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;
```

O Direct Discovery não suporta o uso de agregações em comandos **LOAD**. Se agregações forem utilizadas, os resultados são imprevisíveis. Um comando **LOAD** como o mostrado a seguir não deve ser usado:

```
DIRECT QUERY DIMENSION stateid, SUM(amount*7) AS MultiFirst MEASURE amount FROM sales_table;
```

A **SUM** não deve estar no comando **LOAD**.

O Direct Discovery também não suporta funções do Qlik Sense em comandos **Direct Query**. Por exemplo, a seguinte especificação para um campo **DIMENSION** resulta em uma falha quando o campo "Mth" é usado como uma dimensão em uma visualização:

```
month(ModifiedDate) as Mth
```

Directory

O comando **Directory** define o diretório para procurar os arquivos de dados em comandos **LOAD** subsequentes, até que um novo comando **Directory** seja feito.

Sintaxe:

```
Directory[path]
```

Se o comando **Directory** for emitido sem um **path** ou deixado de fora, o Qlik Sense procurará no diretório de trabalho do Qlik Sense.

Argumentos:

Argumentos

Argumento	Descrição
path	Um texto pode ser interpretado como o caminho para o arquivo data. O caminho é o caminho do arquivo, que pode ser: • absoluto **Exemplo: *c:\data*** • relativo ao diretório operacional do aplicativo Qlik Sense. **Exemplo: *data*** • Endereço de URL (HTTP ou FTP) que aponte para uma localização na Internet ou em uma intranet. **Exemplo: *http://www.qlik.com***

Exemplos:

```
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

O comando **Disconnect** termina a conexão ODBC/OLE DB/Personalizada atual. Esse comando é opcional.

Sintaxe:

```
Disconnect
```

A conexão será encerrada automaticamente quando uma nova declaração **connect** for executada ou quando a execução do script terminar.

Exemplo:

```
Disconnect;
```

Drop

A palavra-chave do script **Drop** pode ser usada para soltar tabelas ou campos do banco de dados.

Drop field

É possível descartar um ou vários campos do Qlik Sense do modelo de dados, e consequentemente da memória, a qualquer momento durante a execução do script usando um comando **drop field**. A propriedade "distinct" de uma tabela é removida após uma declaração **drop field**.

*Tanto **drop field** quanto **drop fields** são formas permitidas e não apresentam diferença no efeito. Se nenhuma tabela for especificada, o campo será descartado de todas as tabelas onde ele ocorre.*

Sintaxe:

```
Drop field fieldname { , fieldname2 ...} [from tablename1 { , tablename2
...}]
Drop fields fieldname { , fieldname2 ...} [from tablename1 { , tablename2
...}]
```

Exemplos:

```
Drop field A;
Drop fields A,B;
Drop field A from X;
Drop fields A,B from X,Y;
```

Drop table

É possível descartar uma ou várias tabelas internas do Qlik Sense do modelo de dados, e consequentemente da memória, a qualquer momento durante a execução do script usando um comando **drop table**.

Sintaxe:

```
drop table  tablename {, tablename2 ...}
drop tables tablename {, tablename2 ...}
```

*As formas **drop table** e **drop tables** são aceitas.*

Os itens a seguir serão perdidos como consequência disso:

- A(s) tabela(s) de fato.
- Todos os campos que não fazem parte das tabelas remanescentes.
- Os valores nos campos restantes, provenientes exclusivamente da(s) tabela(s) descartada(s).

Exemplos e resultados:

Exemplo	Resultado
`drop table Orders, Salesmen, T456a;`	Essa linha resulta em três tabelas descartadas da memória.

`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;`
	Quando uma tabela *Tab2* é criada, a tabela *Tab1* será descartada.

Drop table

É possível descartar uma ou várias tabelas internas do Qlik Sense do modelo de dados, e consequentemente da memória, a qualquer momento durante a execução do script usando um comando **drop table**.

Sintaxe:

```
drop table  tablename {, tablename2 ...}
drop tables tablename {, tablename2 ...}
```

*As formas **drop table** e **drop tables** são aceitas.*

Os itens a seguir serão perdidos como consequência disso:

- A(s) tabela(s) de fato.
- Todos os campos que não fazem parte das tabelas remanescentes.
- Os valores nos campos restantes, provenientes exclusivamente da(s) tabela(s) descartada(s).

Exemplos e resultados:

Exemplo	Resultado
`drop table Orders, Salesmen, T456a;`	Essa linha resulta em três tabelas descartadas da memória.

Exemplo	Resultado
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;	Quando uma tabela *Tab2* é criada, a tabela *Tab1* será descartada.

Execute

O comando **Execute** é utilizado para executar outros programas durante o carregamento de dados do Qlik Sense. Por exemplo, para fazer conversões que sejam necessárias.

Esta funcionalidade não está disponível no Qlik Sense SaaS.

Este comando não é suportado no modo padrão.

Sintaxe:

```
execute commandline
```

Argumentos:

Argumentos

Argumento	Descrição
commandline	Um texto que pode ser interpretado pelo sistema operacional como uma linha de comandos. Você pode consultar um caminho de arquivo absoluto ou um caminho de pasta lib://.

Se desejar usar **Execute**, as seguintes condições devem ser atendidas:

- É necessário executar em modo legado (aplicável para o Qlik Sense e o Qlik Sense Desktop).
- É necessário definir OverrideScriptSecurity para 1 na *Settings.ini* (aplicável para Qlik Sense). *Settings.ini* está localizado em *C:\ProgramData\Qlik\Sense\Engine* e é geralmente um arquivo vazio.

*Se você definir OverrideScriptSecurity para ativar **Execute**, qualquer usuário pode executar arquivos no servidor. Por exemplo, um usuário pode anexar um arquivo executável ao aplicativo, e depois executar o arquivo no script de carregamento de dados.*

Faça o seguinte:

1. Faça uma cópia de *Settings.ini* e abra-o em um editor de texto.
2. Verifique se o arquivo inclui *[Settings 7]* na primeira linha.
3. Insira uma nova linha e digite *OverrideScriptSecurity=1*.
4. Insira uma linha vazia no final do arquivo.
5. Salve o arquivo.
6. Substitua *Settings.ini* pelo arquivo editado.
7. Reinicie o Qlik Sense Engine Service (QES).

Se o Qlik Sense estiver sendo executado como um serviço, alguns comandos podem não funcionar como esperado.

Exemplo:

```
Execute C:\Program Files\Office12\Excel.exe;
Execute lib://win\notepad.exe // win is a folder connection referring to c:\windows
```

Field/Fields

As palavras chaves de script **Field** e **Fields** são usados nos comandos **Declare**, **Derive**, **Drop**, **Comment**, **Rename** e **Tag/Untag**.

FlushLog

O comando **FlushLog** força o Qlik Sense a escrever o conteúdo do buffer do script em um arquivo de log do script.

Sintaxe:

```
FlushLog
```

O conteúdo do buffer é gravado no arquivo de log. Esse comando pode ser útil para fins de depuração, porque você receberá os dados que, de outra forma, podem ter sido perdidos em uma execução de script que falhou.

Exemplo:

```
FlushLog;
```

Force

O comando **force** força o Qlik Sense a interpretar os nomes de campo e valores de campo de comandos **LOAD** e **SELECT** subsequentes como se estivessem escritos apenas em letras

maiúsculas, apenas em letras minúsculas, sempre em maiúsculas ou como são exibidos (em letras maiúsculas e minúsculas). Esse comando permite associar os valores de campo das tabelas criadas de acordo com convenções diferentes.

Sintaxe:

```
Force ( capitalization | case upper | case lower | case mixed )
```

Se nada for especificado, Force Case Mixed será assumido. O comando force será válido até que um novo comando force seja criado.

O comando **force** não tem efeito na seção de acesso: todos os valores de campo carregados não diferenciam maiúsculas de minúsculas.

Exemplos e resultados

Exemplo	**Resultado**

Este exemplo mostra como forçar maiúsculas.
`FORCE Capitalization;`
`Capitalization:`
`LOAD * Inline [`
`ab`
`Cd`
`eF`
`GH`
`];`
	A tabela **Capitalization** contém os seguintes valores:
	`Ab`
	`Cd`
	`Ef`
	`Gh`
	Todos os valores são escritos em maiúsculas.

Este exemplo mostra como forçar a caixa alta.
`FORCE Case Upper;`
`CaseUpper:`
`LOAD * Inline [`
`ab`
`Cd`
`eF`
`GH`
`];`
	A tabela **CaseUpper** contém os seguintes valores:
	`AB`
	`CD`
	`EF`
	`GH`
	Todos os valores estão em caixa alta.

Exemplo	Resultado

Este exemplo mostra como forçar a caixa baixa.
`FORCE Case Lower;`
`CaseLower:`
`LOAD * Inline [`
`ab`
`Cd`
`eF`
`GH`
`];`
	A tabela **CaseLower** contém os seguintes valores:
	`ab`
	`cd`
	`ef`
	`gh`
	Todos os valores estão em caixa baixa.

Este exemplo mostra como forçar a caixa maiúsculas e minúsculas.
`FORCE Case Mixed;`
`CaseMixed:`
`LOAD * Inline [`
`ab`
`Cd`
`eF`
`GH`
`];`
	A tabela **CaseMixed** contém os seguintes valores:
	`ab`
	`Cd`
	`eF`
	`GH`
	Todos os valores são como aparecem no script.

Consulte também:

From

A palavra-chave de script **From** é usada nos comandos de **Load** para se referir a um arquivo e, nos comandos **Select**, para se referir a uma tabela de banco de dados ou exibição.

Load

A declaração **LOAD** carrega campos de um arquivo, de dados definidos no script, de uma tabela de entrada carregada anteriormente, de uma página da Web, do resultado de um comando **SELECT** subsequente ou gerando dados automaticamente. Também é possível carregar dados de conexões analíticas.

Sintaxe:

```
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 ]
```

Argumentos:

Argumentos

Argumento	Descrição
distinct	Você pode usar **distinct** como predicado se deseja carregar apenas registros exclusivos. Se houver registros duplicados, a primeira instância será carregada. Se você estiver usando cargas anteriores, precisará colocar **distinct** na primeira instrução de carga, pois **distinct** afeta apenas a tabela de destino.

Argumento	Descrição
fieldlist	*fieldlist ::= (* | field{**,** * | field })* Uma lista dos campos a serem carregados. O uso de * como lista de campos indica todos os campos da tabela. *field ::= (fieldref | expression) [***as** *aliasname]* A definição de campo deve conter sempre um literal, uma referência a um campo existente ou a uma expressão. *fieldref ::= (fieldname |@fieldnumber |@startpos:endpos [* ***I*** | ***U*** | ***R*** | ***B*** | **T**]) *fieldname* é um texto idêntico a um nome de campo da tabela. Observe que o nome do campo deverá estar entre aspas duplas ou colchetes se contiver espaços, por exemplo. Às vezes, os nomes de campo não estão explicitamente disponíveis. Assim, uma notação diferente será usada: *@fieldnumber* representa o número de campo em um arquivo de tabela delimitado. Deve ser um inteiro positivo, precedido de “@”. A numeração sempre inicia no número 1 até o número de campos. *@startpos*:*endpos* representa as posições inicial e final de um campo em um arquivo com registros de comprimento fixo. As posições devem ser inteiros positivos. Os dois números devem ser precedidos de "@" e separados por dois-pontos. A numeração sempre inicia no número 1 até o número de posições. No último campo, **n** é usado como posição final do campo. • Se *@startpos*:*endpos* for seguido imediatamente pelos caracteres **I** ou **U**, os bytes lidos serão interpretados como um inteiro binário assinado (**I**) ou não assinado (**U**) (ordem dos bytes da Intel). O número das posições lidas deve ser 1, 2 ou 4. • Se *@startpos*:*endpos* for imediatamente seguido pelo caractere **R**, os bytes lidos serão interpretados como um número real binário (ponto flutuante IEEE de 32 ou de 64 bits). O número das posições lidas deve ser 4 ou 8. • Se *@startpos*:*endpos* for seguido imediatamente pelo caractere **B**, os bytes lidos serão interpretados como números BCD (Binary Coded Decimal), de acordo com o padrão COMP-3. Qualquer número de bytes pode ser especificado. *expression* pode ser uma função numérica ou uma função de string baseada em um ou vários outros campos da mesma tabela. Para obter mais informações, consulte a sintaxe das expressões. **as** é usado para atribuir um novo nome ao campo.

Argumento	Descrição
from	**from** é usado se os dados precisam ser carregados a partir de um arquivo usando uma pasta ou uma conexão de dados de um arquivo da web *file ::= [path] filename* **Exemplo: *'lib://Table Files/'*** Quando o caminho for omitido, o Qlik Sense procura o arquivo no diretório especificado pelo comando **Directory**. Se não houver um comando **Directory**, o Qlik Sense pesquisará no diretório de trabalho, *C:\Users\{user}\Documents\Qlik\Sense\Apps*. *Em uma instalação de servidor do Qlik Sense, o diretório de trabalho é especificado no Qlik Sense Repository Service; por padrão é C:\ProgramData\Qlik\Sense\Apps.* O *filename* pode conter os caracteres curingas padrão do DOS (* e ?). Isso carregará todos os arquivos correspondentes no diretório especificado. *format-spec ::= (fspec-item { , fspec-item })* A especificação do formato consiste em uma lista de vários itens de especificação de formato, entre colchetes. **Modo de script legado** No modo legado de uso de scripts, os seguintes formatos de caminho também são suportados: • absoluto **Exemplo: *c:\data*** • relativo ao diretório operacional do aplicativo Qlik Sense. **Exemplo: *data*** • Endereço de URL (HTTP ou FTP) que aponte para uma localização na Internet ou em uma intranet. **Exemplo: *http://www.qlik.com*** •

Argumento	Descrição
from_field	**from_field** é usado se for necessário carregar dados de um campo anteriormente carregado. *fieldassource::=***(***tablename, fieldname****)*** O campo é o nome de *tablename* e *fieldname* carregados anteriormente. *format-spec ::= (fspec-item {, fspec-item })* A especificação do formato consiste em uma lista de vários itens de especificação de formato, entre colchetes. Para obter mais informações, consulte *Itens de especificação de formato (page 167)*. ***from_field*** *oferece suporte apenas a vírgulas como delimitador de lista ao separar campos em tabelas.*
inline	**inline** será usado se os dados precisarem ser digitados no script, e não carregados de um arquivo. *data ::= [text]* Os dados inseridos por uma cláusula **inline** devem ser colocados entre aspas duplas ou colchetes. O texto entre esses elementos será interpretado da mesma maneira que o conteúdo de um arquivo. Portanto, no local em que uma nova linha for inserida em um arquivo de texto, você também deverá inseri-la no texto de uma cláusula **inline**, isto é, pressionando a tecla Enter ao digitar o script. O número de colunas é definido pela primeira linha. *format-spec ::= (fspec-item {, fspec-item })* A especificação do formato consiste em uma lista de vários itens de especificação de formato, entre colchetes. Para obter mais informações, consulte *Itens de especificação de formato (page 167)*.
resident	**resident** é usado se for necessário carregar dados de uma tabela anteriormente carregada. *table label* é um rótulo que precede o comando **LOAD** ou **SELECT** que criou a tabela original. O rótulo deve ter dois-pontos no final.
autogenerate	**autogenerate** será usado se for necessário gerar os dados automaticamente pelo Qlik Sense. *size ::= number* *Number* é um inteiro que indica o número de registros a serem gerados. A lista de campos não deve conter expressões que precisem de dados de uma fonte de dados externa ou de uma tabela carregada anteriormente, a menos que você faça referência a um único valor de campo em uma tabela carregada anteriormente com a função **Peek**.

Argumento	Descrição
extension	É possível carregar dados de conexões analíticas. Você precisa usar a cláusula **extension** para chamar uma função definida no plug-in SSE (Server-Side Extension) ou avaliar um script. Você pode enviar uma única tabela ao plug-in SSE, e uma única tabela de dados é retornada. Se o plug-in não especificar os nomes dos campos que são retornados, os campos serão nomeados como Field1, Field2, e assim por diante. `Extension pluginname.functionname( tabledescription );` • Carregando dados com o uso de uma função em um plug-in SSE *tabledescription ::= (table { ,tablefield})* Se você não indicar campos de tabela, os campos serão usados na ordem de carregamento. • Carregando dados ao avaliar um script em um plug-in SSE *tabledescription ::= (script, table { ,tablefield})* **Manipulação de tipos de dados na definição de campo de tabela** Tipos de dados são detectados automaticamente em conexões analíticas. Se os dados não tiverem valores numéricos e pelo menos uma string de texto não NULL, o campo será considerado texto. Em qualquer outro caso, ele será considerado numérico. Você pode forçar o tipo de dados envolvendo um nome de campo com **String()** ou **Mixed()**. • **String()** força o campo como texto. Se o campo for numérico, a parte de texto do valor duplo será extraída, e nenhuma conversão será realizada. • **Mixed()** força o campo como duplo. **String()** ou **Mixed()** não podem ser usados fora de definições de campo de tabela **extension**, e você não pode usar outras funções Qlik Sense em uma definição de campo de tabela. **Mais sobre conexões analíticas** Você precisa configurar conexões analíticas antes de poder usá-las.
where	**where** é uma cláusula utilizada para declarar se um registro deve ou não ser incluído na seleção. A seleção será incluída se *criterion* for True. *criterion* é uma expressão lógica.
while	**while** é uma cláusula usada para declarar que um registro deve ser lido repetidamente. O mesmo registro será lido se *criterion* forTrue. Para que possa ser útil, a cláusula **while** normalmente deve incluir a função **IterNo( )**. *criterion* é uma expressão lógica.

Argumento	Descrição
group by	**group by** é uma cláusula usada para definir os campos de agregação (agrupamento) dos dados. Os campos de agregação devem ser incluídos de alguma maneira nas expressões carregadas. Nenhum outro campo diferente desses poderá ser usado fora das funções de agregação nas expressões carregadas. *groupbyfieldlist ::= (fieldname { ,fieldname })*
order by	**order by** é uma cláusula usada para classificar os registros de uma tabela residente antes de seu processamento pela declaração **load**. A tabela residente pode ser classificada por um ou mais campos, em ordem ascendente ou descendente. A classificação é feita, principalmente, pelo valor numérico e, em seguida, pela ordem nacional de colação. Essa cláusula somente poderá ser usada quando a fonte de dados for uma tabela residente. Os campos de classificação especificam por qual campo a tabela residente é classificada. O campo pode ser especificado por seu nome ou número na tabela residente (o primeiro campo é o número 1). orderbyfieldlist ::= fieldname [sortorder] { , fieldname [sortorder] } *sortorder* é *asc* para ascendente ou *desc* para descendente. Se nenhuma *sortorder* for especificada, *asc* será assumido. *fieldname*, *path*, *filename* e *aliasname* são strings de texto que indicam o que seus respectivos nomes implicitamente representam. Qualquer campo na tabela de origem pode ser usado como *fieldname*. No entanto, campos criados por meio da cláusula (*aliasname*) estão fora de escopo e não podem ser usados dentro de uma mesma declaração **load**.

Se nenhuma fonte de dados for fornecida por meio de uma cláusula **from**, **inline**, **resident**, **from_field,** **extension** ou **autogenerate**, os dados serão carregados a partir do resultado imediatamente após a instrução **SELECT** ou **LOAD**. O comando sucessivo não deverá ter prefixo.

Exemplos:

Carregando diferentes formatos de arquivo
Carregue um arquivo de dados delimitados com opções padrão:

```
LOAD * from data1.csv;
```

Carregar um arquivo de dados delimitados a partir de uma conexão da biblioteca (DataFiles):

```
LOAD * from 'lib://DataFiles/data1.csv';
```

Carregar todos os arquivos de dados delimitados a partir de uma conexão da biblioteca (DataFiles):

```
LOAD * from 'lib://DataFiles/*.csv';
```

Carregue um arquivo delimitado, especificando vírgula como separador e com rótulos embutidos:

```
LOAD * from 'c:\userfiles\data1.csv' (ansi, txt, delimiter is ',', embedded labels);
```

Carregue um arquivo delimitado, especificando a guia como separador e com rótulos embutidos:

```
LOAD * from 'c:\userfiles\data2.txt' (ansi, txt, delimiter is '\t', embedded labels);
```

Carregue um arquivo dif com cabeçalhos integrados:

```
LOAD * from file2.dif (ansi, dif, embedded labels);
```

Carregue três campos de um arquivo de registro fixo sem cabeçalhos:

```
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);
```

Carregue um arquivo QVX, especificando um caminho absoluto:

```
LOAD * from C:\qdssamples\xyz.qvx (qvx);
```

Carregando arquivo da Web
Carregue a partir da URL padrão definida na conexão de dados do arquivo da Web:

```
LOAD * from [lib://MyWebFile];
```

Carregue a partir de uma URL específica e substitua a URL definida na conexão de dados do arquivo da Web:

```
LOAD * from [lib://MyWebFile] (URL is 'http://localhost:8000/foo.bar');
```

Carregue a partir de uma URL específica definida em uma variável usando a expansão de sinal de dólar:

```
SET dynamicURL = 'http://localhost/foo.bar';

LOAD * from [lib://MyWebFile] (URL is '$(dynamicURL)');
```

Selecionando campos específicos, renomeando e calculando campos
Carregue somente três campos específicos de um arquivo delimitado:

```
LOAD FirstName, LastName, Number from data1.csv;
```

Renomeie o primeiro campo como A e o segundo campo como B ao carregar um arquivo sem rótulos:

```
LOAD @1 as A, @2 as B from data3.txt (ansi, txt, delimiter is '\t', no labels);
```

Carregue Name como uma concatenação de FirstName, um caractere de espaço e LastName:

```
LOAD FirstName&' '&LastName as Name from data1.csv;
```

Carregue Quantity, Price e Value (o produto de Quantity e Price):

```
LOAD Quantity, Price, Quantity*Price as Value from data1.csv;
```

Selecionando registros específicos
Carregue somente registros exclusivos, e os registros duplicados serão descarregados:

```
LOAD distinct FirstName, LastName, Number from data1.csv;
```

Carregue somente registros onde o campo Litres tiver um valor acima de zero:

```
LOAD * from Consumption.csv where Litres>0;
```

Carregando dados que não estão no arquivo e dados gerados automaticamente
Carregue uma tabela com dados inline, dois campos nomeados CatID e Category:

```
LOAD * Inline

[CatID, Category

0,Regular

1,Occasional

2,Permanent];
```

Carregue uma tabela com dados inline, três campos nomeados UserID, Password e Access:

```
LOAD * Inline [UserID, Password, Access

A, ABC456, User

B, VIP789, Admin];
```

Carregue uma tabela com 10 mil linhas. O campo A irá conter o número do registro de leitura (1,2,3,4,5...) e o campo B irá conter um número aleatório entre 0 e 1:

```
LOAD RecNo( ) as A, rand( ) as B autogenerate(10000);
```

Os parênteses após autogenerate são permitidos, mas não obrigatórios.

Carregando dados de uma tabela carregada anteriormente
Primeiro nós carregamos um arquivo de tabela delimitado e o nomeamos tab1:

```
tab1:

SELECT A,B,C,D from 'lib://DataFiles/data1.csv';
```

Carregue campos da tabela já carregada tab1 como tab2:

```
tab2:

LOAD A,B,month(C),A*B+D as E resident tab1;
```

Carregue campos da tabela já carregada tab1, mas somente registros onde A seja maior que B:

```
tab3:
```

```
LOAD A,A+B+C resident tab1 where A>B;
```

Carregue campos da tabela já carregada tab1 ordenada por A:

```
LOAD A,B*C as E resident tab1 order by A;
```

Carregue campos da tabela já carregada tab1, ordenada pelo primeiro campo e, em seguida, o segundo campo:

```
LOAD A,B*C as E resident tab1 order by 1,2;
```

Carregue campos da tabela já carregada tab1 ordenada por C decrescente e B em ordem crescente e, em seguida, o primeiro campo em ordem decrescente:

```
LOAD A,B*C as E resident tab1 order by C desc, B asc, 1 desc;
```

Carregando dados de campos carregados anteriormente
Carregue o campo Types da tabela carregada anteriormente Characters como A:

```
LOAD A from_field (Characters, Types);
```

Carregando dados de uma tabela posterior (carga anterior
Carregue A, B e os campos calculados X e Y do Table1 que é carregado na instrução **SELECT** seguinte:

```
LOAD A, B, if(C>0,'positive','negative') as X, weekday(D) as Y;

SELECT A,B,C,D from Table1;
```

Dados de agrupamento
Campos de carga agrupados (agregados) por ArtNo:

```
LOAD ArtNo, round(Sum(TransAmount),0.05) as ArtNoTotal from table.csv group by ArtNo;
```

Campos de carga agrupados (agregados) por Week e ArtNo:

```
LOAD Week, ArtNo, round(Avg(TransAmount),0.05) as WeekArtNoAverages from table.csv group by
Week, ArtNo;
```

Lendo um registro repetidamente
Nesse exemplo temos um arquivo de entrada Grades.csv que contém as notas para cada aluno condensadas em um campo:

```
Student,Grades

Mike,5234
```

```
John,3345

Pete,1234

Paul,3352
```

As notas, em uma escala de 1-5, representam as matérias Math, English, Science e History. Podemos separar as notas em valores separados, lendo cada registro várias vezes com uma cláusula **while**, utilizando a função **IterNo( )** como um contador. Em cada leitura, a nota é extraída com a função **Mid** e armazenada em Grade. A matéria é selecionada usando a função **pick** e armazenada em Subject. A cláusula final **while** contém o teste para verificar se todas as notas foram lidas (quatro por aluno, neste caso), o que significa que o registro do próximo aluno deve ser lido.

```
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));
```

O resultado é uma tabela que contém estes dados:

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

Carregando de conexões analíticas

Os seguintes dados de amostra são usados.

```
Values:
Load
 Rand() as A,
 Rand() as B,
 Rand() as C
AutoGenerate(50);
```

Carregando dados usando uma função

Nestes exemplos, partimos do princípio de que temos um plug-in de conexão analítica *P* que contém uma função *Calculate(Parameter1, Parameter2)*. A função retorna a tabela *Resultados*, que contém os campos *Field1* e *Field2*.

```
Load * Extension P.Calculate( Values{A, C} );
```
Carregue todos os campos retornados ao enviar os campos A e C para a função.

```
Load Field1 Extension P.Calculate( Values{A, C} );
```
Carregue apenas o campo Field1 ao enviar os campos A e C para a função.

```
Load * Extension P.Calculate( Values );
```
Carregue todos os campos retornados ao enviar os campos A e B para a função. Como campos não estão especificados, A e B são usados, pois são os primeiros na ordem da tabela.

```
Load * Extension P.Calculate( Values {C, C});
```
Carregue todos os campos que são retornados ao enviar o campo C a ambos os parâmetros da função.

```
Load * Extension P.Calculate( Values {String(A), Mixed(B)});
```
Carregue todos os campos que são retornados ao enviar o campo A forçado como uma string e B forçado como um número para a função.

Carregando dados ao avaliar um script

```
Load A as A_echo, B as B_echo Extension R.ScriptEval( 'q;', Values{A, B} );
```
Carregue a tabela retornada pelo script q ao enviar os valores de A e B.

```
Load * Extension R.ScriptEval( '$(My_R_Script)', Values{A, B} );
```
Carregue a tabela retornada pelo script armazenado na variável My_R_Script ao enviar os valores de A e B.

```
Load * Extension R.ScriptEval( '$(My_R_Script)', Values{B as D, *} );
```
Carregue a tabela retornada pelo script armazenado na variável My_R_Script ao enviar os valores de B renomeados como D, A e C. Usar * envia os campos restantes não referenciados.

A extensão do arquivo de conexões DataFiles faz distinção entre maiúsculas e minúsculas. Por exemplo: .qvd.

Itens de especificação de formato

Cada item de especificação de formato define uma propriedade específica do arquivo de tabela:

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`]

Conjunto de caracteres

Conjunto de caracteres é um especificador de arquivo do comando **LOAD** que define o conjunto de caracteres usado no arquivo.

Os especificadores **ansi**, **oem** e **mac** foram usados no QlikView e continuarão funcionando. No entanto, eles não serão gerados durante a criação do comando **LOAD** com o Qlik Sense.

Sintaxe:

```
utf8 | unicode | ansi | oem | mac | codepage is
```

Argumentos:

Argumentos

Argumento	Descrição
utf8	Conjunto de caracteres UTF-8
unicode	Conjunto de caracteres Unicode
ansi	Windows, codepage 1252
oem	DOS, OS/2, AS400 e outros
mac	Codepage 10000
codepage is	Com o especificador **codepage** , é possível usar qualquer codepage do Windows como *N* .

Limitações:

A conversão do conjunto de caracteres **oem** não é implementada para o macOS. Se nada for especificado, o codepage 1252 será assumido no Windows.

Exemplo:

```
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)
```

Consulte também:

Load (page 157)

Formato da tabela

O formato da tabela é um especificador de arquivo do comando **LOAD** que define o tipo de arquivo. Se não houver especificação, um arquivo *.txt* será adotado.

Tipos de formatos de tabela

Tipo	Descrição
txt	Em um arquivo de texto delimitado, as colunas na tabela são separadas por um caractere separador.
fix	Em um arquivo de registro fixo, cada campo tem um determinado número de caracteres. Geralmente, muitos arquivos de comprimento de registro fixo contêm registros separados por um avanço de linha, mas existem opções mais avançadas para especificar o tamanho do registro em bytes ou para alcançar mais de uma linha com **Record is**. *Se os dados contiverem caracteres com vários bytes, as interrupções de campo poderão ficar desalinhadas, já que o formato é baseado em um comprimento fixo em bytes.*
dif	Em um arquivo *.dif* (Data Interchange Format), é utilizado um formato especial para a definição da tabela.
biff	OQlik Sense também pode interpretar dados em arquivos padrão do Excel por meio do formato *biff* (Binary Interchange File Format).
ooxml	O Excel 2007 e as versões mais recentes usam o formato ooxml *.xslx*.
html	Se a tabela fizer parte de uma página ou arquivo html, o html deverá ser usado.
xml	xml (Extensible Markup Language) é uma linguagem de marcação comum que é usada para representar as estruturas de dados em um formato de texto.
qvd	O formato *qvd* é proprietário dos arquivos QVD exportado do Qlik Sense Qlik Cloud.
qvx	*qvx* é um formato de arquivo/fluxo para saídas de alto desempenho do Qlik Sense.
parquet	Apache Parquet é um formato de armazenamento em colunas, altamente eficiente para armazenar e consultar conjuntos de dados grandes.

Delimiter is

Para arquivos de tabela delimitada, é possível especificar um separador arbitrário usando a especificação **delimiter is**. Essa especificação é relevante apenas para arquivos .txt delimitados.

Sintaxe:

```
delimiter is char
```

Argumentos:

Argumentos

Argumento	Descrição
char	Especifica um único caractere a partir dos caracteres 127 ASCII.

Além disso, podem ser usados os seguintes valores:

Valores opcionais

Valor	Descrição
'\t'	representa um sinal de aba, com ou sem sinais de aspas.
'\\'	representa um caractere de barra invertida (\ ).
'spaces'	representa todas as combinações de um ou mais espaços. Os caracteres não imprimíveis com valor ASCII abaixo de 32, com exceção de CR e LF, serão interpretados como espaços.

Se nada for especificado, **delimiter is ','** será assumido.

Exemplo:

```
LOAD * from a.txt (utf8, txt, delimiter is ',' , embedded labels);
```

Consulte também:

Load (page 157)

No eof

A especificação **no eof** é usada para ignorar caracteres de fim de arquivo ao carregar arquivos **.txt** delimitados.

Sintaxe:

```
no eof
```

Se o especificador **no eof** for usado, caracteres com ponto de código 26, que em outros contextos denotam o final do arquivo, serão desconsiderados e poderão fazer parte de um valor de campo.

É relevante apenas para arquivos de texto delimitados.

Exemplo:

```
LOAD * from a.txt (txt, utf8, embedded labels, delimiter is ' ', no eof);
```

Consulte também:

Load (page 157)

Labels

Labels é uma especificação de arquivo para a declaração **LOAD** que define onde os nomes de arquivo podem ser encontrados em um arquivo.

Sintaxe:

```
embedded labels|explicit labels|no labels
```

Os nomes de campo podem estar localizados em diferentes partes no arquivo. Se o primeiro registro contiver os nomes dos campos, **embedded labels** deverá ser usado. Se não existirem nomes de campo, **no labels** deverá ser usado. Nos arquivos *dif*, algumas vezes uma seção de cabeçalho separada é usada com nomes de campo explícitos. Nesse caso, **explicit labels** deverá ser usado. Se nada for especificado, **embedded labels** será assumido também para arquivos *dif*.

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)
```

Consulte também:

Load (page 157)

Header is

Especifica o tamanho de cabeçalho em arquivos de tabela. Um comprimento de cabeçalho arbitrário pode ser especificado com a especificação **header is**. O cabeçalho é uma seção de texto não usada pelo Qlik Sense.

Sintaxe:

```
header is n
```

```
header is line
```

```
header is n lines
```

O comprimento do cabeçalho pode ser fornecido em bytes (**header is n**) ou em linhas (**header is line** ou **header is n lines**). **n** deve ser um inteiro positivo, representando o comprimento do cabeçalho. Se não for especificado, **header is 0** será assumido. O especificador **header is** é relevante apenas para arquivos de tabela.

Exemplo:

Esse é um exemplo de uma tabela de fonte de dados contendo uma linha de texto de cabeçalho que não deve ser interpretada como um dado do Qlik Sense.

```
*Header line
Col1,Col2
a,B
c,D
```

Usando o especificador **header is 1 lines**, a primeira linha não será carregada como dados. No exemplo, o especificador **embedded labels** informa ao Qlik Sense que a primeira linha não excluída seja interpretada como contendo rótulos de campo.

```
LOAD Col1, Col2
FROM 'lib://files/header.txt'
(txt, embedded labels, delimiter is ',', msq, header is 1 lines);
```

O resultado é uma tabela com dois campos, Col1 e Col2.

Consulte também:

Load (page 157)

Record is

Para arquivos de comprimento de registro fixo, o comprimento do registro deve ser estabelecido por meio da especificação **record is**.

Sintaxe:

```
Record is n
Record is line
Record is n lines
```

Argumentos:

Argumentos

Argumento	Descrição
n	Especifica o comprimento do registro em bytes.
line	Especifica o comprimento do registro como uma linha.
n lines	Especifica o comprimento do registro em linhas, em que n é um inteiro positivo representando o comprimento do registro.

Limitações:

O especificador **record is** é relevante apenas para arquivos **fix**.

Consulte também:

Load (page 157)

Quotes

Quotes são um especificador de arquivo do comando **LOAD** que define se as aspas podem ser usadas e a precedência entre aspas e separadores. Para arquivos de texto apenas.

Sintaxe:

```
no quotes
```

```
msq
```

Se o especificador for omitido, as aspas padrão serão usadas, isto é, as aspas " " ou ' ' poderão ser usadas, mas somente se forem o primeiro e o último caractere não em branco do valor de um campo.

Argumentos:

Argumentos

Argumento	Descrição
no quotes	Usado se as aspas não forem aceitas em um arquivo de texto.
msq	Usado para especificar aspas de estilo moderno, o que permite conteúdo em várias linhas em campos. Campos contendo caracteres de final de linha devem ser colocados entre aspas duplas. Uma limitação do especificador msq é que os caracteres de aspas duplas (") aparecendo como o primeiro ou o último caractere no conteúdo do campo serão interpretados como início ou fim de conteúdo em várias linhas, o que pode acarretar resultados imprevisíveis no conjunto de dados carregado. Nesse caso, você deve usar aspas padrão, omitindo o especificador.

XML

Essa especificação de script é usada ao carregar arquivos xml. As opções válidas para o especificador **XML** estão listadas na sintaxe.

Não é possível carregar arquivos DTD no Qlik Sense.

Sintaxe:

```
xmlsimple
```

Consulte também:

Load (page 157)

KML

O especificador de script é usado ao carregar o arquivoKML para usar em uma visualização de mapa.

Sintaxe:

```
kml
```

O arquivo KML pode representar dados de área (por exemplo, países ou regiões) representados por polígonos, dados de linhas (por exemplo, pistas ou estradas) ou dados de pontos (por exemplo, cidades ou locais) representados por pontos no formato [long, lat].

URL is

Esse especificador de script é usado para definir a URL de uma conexão de dados de arquivo da Web ao carregar um arquivo da Web.

Sintaxe:

```
URL is string
```

Argumentos:

Argumentos

Argumento	Descrição
string	Especifica a URL do arquivo a ser carregado. Isso substituirá a URL definida na conexão de arquivo da Web que é usada.

Limitações:

O especificador **URL is** é apenas relevante para arquivos da Web. Você precisa usar uma conexão de dados de arquivo da Web existente.

Consulte também:

Load (page 157)

userAgent is

Esse especificador de script é usado para definir o agente usuário do navegador ao carregar um arquivo da Web.

Sintaxe:

```
userAgent is string
```

Argumentos:

Argumentos

Argumento	Descrição
string	Especifica a string do agente usuário do navegador. Isso substituirá o agente usuário do navegador padrão "Mozilla/5.0".

Limitações:

O especificador **userAgent is** é apenas relevante para arquivos da Web.

Consulte também:

Load (page 157)

Let

O comando **let** é um complemento ao comando **set**, usado para definir variáveis de script. O comando **let**, ao contrário do comando **set**, avalia a expressão no lado direito do sinal de igual "=" no tempo de execução do script antes de ser atribuída à variável.

Sintaxe:

```
Let variablename=expression
```

Exemplos e resultados:

Exemplo	Resultado

`Set x=3+4;`
 	
`Let y=3+4;`
 	
`z=$(y)+1;`
	`$(x)` será avaliado como ' 3+4 '
	
	`$(y)` será avaliado como ' 7 '
	
	`$(z)` será avaliado como ' 8 '
	
	Observe a diferença entre os comandos **Set** e **Let**. A instrução **Set** atribui a string "3+4" à variável, enquanto a instrução **Let** avalia a string e atribui 7 à variável.

Let T=now( );	`$(T)` receberá o valor da hora atual.

Loosen Table

Uma ou mais tabelas de dados internas do Qlik Sense podem ser declaradas explicitamente como parcialmente desconectadas durante a execução do script usando um comando **Loosen Table**. Quando uma tabela é parcialmente desconectada, todas as associações entre os valores de campo da tabela são removidos. Um efeito semelhante pode ser conseguido por meio do carregamento de cada campo da tabela parcialmente desconectada como tabelas independentes desconectadas. A tabela parcialmente desconectada pode ser útil durante os testes para isolar temporariamente diferentes partes da estrutura de dados. Uma tabela parcialmente desconectada pode ser identificada pelas linhas pontilhadas no visualizador de tabelas. O uso de um ou mais comandos **Loosen Table** no script fará com que o Qlik Sense desconsidere qualquer configuração das tabelas como parcialmente desconectadas feita antes da execução do script.

Sintaxe:

```
Loosen Tabletablename [ , tablename2 ...]
```

```
Loosen Tablestablename [ , tablename2 ...]
```

Pode-se utilizar qualquer uma das sintaxes: **Loosen Table** ou **Loosen Tables**.

*Caso o Qlik Sense encontre referências circulares na estrutura dos dados que não possam ser interrompidas, de forma interativa ou explícita no script, pelas tabelas declaradas como parcialmente desconectadas, uma ou mais tabelas adicionais serão forçadas como parcialmente desconectadas até que não haja mais nenhuma referência circular. Quando isso acontecer, o diálogo **Aviso de Referência Circular** fornecerá um aviso.*

Exemplo:

```
Tab1:

SELECT * from Trans;

Loosen Table Tab1;
```

Map

O comando **map ... using** é usado para mapear um determinado valor de campo ou uma expressão para os valores de uma tabela de mapeamento específica. A tabela de mapeamento é criada pelo comando **Mapping**.

Sintaxe:

```
Map fieldlist Using  mapname
```

O mapeamento automático é feito para campos carregados depois do comando **Map ... Using** até o final do script ou até um comando **Unmap** ser encontrado.

O mapeamento ocorre por último na cadeia de eventos e leva ao campo armazenado na tabela interna do Qlik Sense. Isso significa que o mapeamento não é feito sempre que um nome de campo é encontrado como parte de uma expressão, mas quando o valor está armazenado sob o nome de campo na tabela interna. Se o mapeamento em nível da expressão for necessário, a função **Applymap()** deverá ser usada.

Argumentos:

Argumentos

Argumento	Descrição
fieldlist	Uma lista dos campos separados por vírgulas que devem ser mapeados a partir desse ponto no script. O uso de * como lista de campos indica todos os campos. Os caracteres curingas * e ? são permitidos nos nomes de campo. Poderá ser necessário colocar os nomes de campos entre aspas quando forem utilizados caracteres curinga.
mapname	O nome de uma tabela de mapeamento lida anteriormente em um comando **mapping load** ou **mapping select**.

Exemplos e resultados:

Exemplo	Resultado
Map Country Using Cmap;	Permite o mapeamento do campo Country usando o mapa Cmap.
Map A, B, C Using X;	Permite o mapeamento dos campos A, B e C usando o mapa X.
Map * Using GenMap;	Permite o mapeamento de todos os campos usando GenMap.

NullAsNull

O comando **NullAsNull** desativa a conversão de valores NULL em strings anteriormente definidas pela declaração **NullAsValue**.

Sintaxe:

```
NullAsNull *fieldlist
```

O comando**NullAsValue** funciona como uma opção e pode ser ativada ou desativada várias vezes no script por meio de um comando **NullAsValue** ou de um comando **NullAsNull**.

Argumentos:

Argumentos

Argumento	Descrição
*fieldlist	Uma lista separada por vírgula dos campos nos quais **NullAsNull** deve ser ativado. O uso de * como lista de campos indica todos os campos. Os caracteres curingas * e ? são permitidos nos nomes de campo. Poderá ser necessário colocar os nomes de campos entre aspas quando forem utilizados caracteres curinga.

Exemplo:

```
NullAsNull A,B;
LOAD A,B from x.csv;
```

NullAsValue

O comando **NullAsValue** especifica para quais campos os NULL encontrados devem ser convertidos em um valor.

Sintaxe:

```
NullAsValue *fieldlist
```

Por padrão, o Qlik Sense considera os valores NULL como entidades ausentes ou indefinidas. No entanto, em alguns contextos de base de dados, os valores NULL devem ser considerados valores especiais, e não apenas valores ausentes. O fato de os valores NULL normalmente não poderem ser vinculados a outros valores NULL pode ser suspenso por meio do comando **NullAsValue**.

O comando **NullAsValue** opera como uma opção e funcionará em comandos de carregamento subsequentes. Pode ser desativado novamente por meio do comando **NullAsNull**.

Argumentos:

Argumentos

Argumento	Descrição
*fieldlist	Uma lista separada por vírgula dos campos nos quais **NullAsValue** deve ser ativado. O uso de * como lista de campos indica todos os campos. Os caracteres curingas * e ? são permitidos nos nomes de campo. Poderá ser necessário colocar os nomes de campos entre aspas quando forem utilizados caracteres curinga.

Exemplo:

```
NullAsValue A,B;
Set NullValue = 'NULL';
LOAD A,B from x.csv;
```

Qualify

O comando **Qualify** é usado para alterar a qualificação dos nomes de campo, ou seja, nomes de campo receberão o nome da tabela como um prefixo.

Sintaxe:

```
Qualify *fieldlist
```

A junção automática entre campos com o mesmo nome em tabelas diferentes pode ser suspensa usando o comando **qualify**, que qualifica o nome do campo com seu nome de tabela. Se for(em) qualificado(s), o(s) nome(s) de campo será(ão) renomeado(s) quando encontrado(s) em uma tabela. O novo nome estará no formato *tablename.fieldname*. *Tablename* é equivalente ao rótulo da tabela atual ou, se não houver rótulo, ao nome que aparece depois de **from** nos comandos **LOAD** e **SELECT**.

A qualificação será feita para todos os campos carregados depois do comando **qualify**.

A qualificação é sempre desativada por padrão no início da execução do script. A qualificação do nome de um campo pode ser ativada a qualquer momento usando um comando **qualify**. A qualificação pode ser desativada a qualquer momento usando um comando **Unqualify**.

*O comando **qualify** não deve ser usado em conjunto com uma recarga parcial.*

Argumentos:

Argumentos

Argumento	Descrição
*fieldlist	Uma lista separada por vírgula dos campos nos quais a qualificação deve ser ativada. O uso de * como lista de campos indica todos os campos. Os caracteres curingas * e ? são permitidos nos nomes de campo. Poderá ser necessário colocar os nomes de campos entre aspas quando forem utilizados caracteres curinga.

Example 1:

```
Qualify B;

LOAD A,B from x.csv;

LOAD A,B from y.csv;
```

As duas tabelas, **x.csv** e **y.csv**, são associadas somente por **A**. Três campos serão resultantes: A, x.B, y.B.

Example 2:

Em uma base de dados desconhecida, costuma ser útil, a princípio, assegurar-se de que apenas um ou poucos campos sejam associados, como ilustrado neste exemplo:

```
qualify *;

unqualify TransID;

SQL SELECT * from tab1;

SQL SELECT * from tab2;

SQL SELECT * from tab3;
```

Somente **TransID** será usado para associações entre as tabelas *tab1*, *tab2* e *tab3*.

Rem

A declaração **rem** é utilizada para inserir comentários no script ou para desativar temporariamente comandos do script sem removê-los.

Sintaxe:

```
Rem string
```

Tudo que estiver entre **rem** e o próximo ponto e vírgula **;** será considerado um comentário.

Há dois métodos alternativos disponíveis para criar comentários no script:

1. É possível criar um comentário em qualquer lugar do script – exceto entre dois sinais de aspas – posicionando-se a seção em questão entre **/*** e ***/**.
2. Ao digitar **//** no script, todo o texto que vem a seguir, à direita, na mesma linha, torna-se um comentário. (Observe que a exceção //: pode ser utilizada como parte de um endereço da Internet).

Argumentos:

Argumentos

Argumento	Descrição
string	Um texto arbitrário.

Exemplo:

```
Rem ** This is a comment **;
/* This is also a comment */
// This is a comment as well
```

Rename

A palavra-chave do script **Rename** pode ser usada para renomear campos ou tabelas que já foram carregadas.

Rename field

Essa função de script renomeia um ou mais campos do Qlik Sense existentes depois que eles foram carregados.

No Qlik Sense, não é recomendado dar a uma variável o nome de um campo ou função.

Pode-se utilizar qualquer uma das sintaxes: **rename field** ou **rename fields**.

Sintaxe:

```
Rename Field (using mapname | oldname to newname{ , oldname to newname })
```

```
Rename Fields (using mapname | oldname to newname{ , oldname to newname })
```

Argumentos:

Argumento	Descrição
mapname	O nome de uma tabela de mapeamento carregada anteriormente, contendo um ou mais pares de nomes de campos antigos e novos.
oldname	O antigo nome do campo.
newname	O novo nome do campo.

Limitações:

Não é possível renomear dois campos para ter o mesmo nome.

Example 1:

```
Rename Field XAZ0007 to Sales;
```

Example 2:

```
FieldMap:

Mapping SQL SELECT oldnames, newnames from datadictionary;

Rename Fields using FieldMap;
```

Rename table

Essa função de script renomeia uma ou mais tabelas internas existentes do Qlik Sense depois que elas foram carregadas.

Pode-se utilizar qualquer uma das sintaxes: **rename table** ou **rename tables**.

Sintaxe:

```
Rename Table (using mapname | oldname to newname{ , oldname to newname })
Rename Tables (using mapname | oldname to newname{ , oldname to newname })
```

Argumentos:

Argumentos

Argumento	Descrição
mapname	O nome de uma tabela de mapeamento carregada anteriormente, contendo um ou mais pares de nomes de tabelas antigos e novos.
oldname	O antigo nome da tabela.
newname	O novo nome da tabela.

Limitações:

Duas tabelas nomeadas de maneira diferente não podem ser renomeadas com o mesmo nome. O script gerará um erro se você tentar renomear uma tabela com o mesmo nome de uma tabela existente.

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

O comando **Search** é utilizado para incluir ou excluir campos na pesquisa inteligente.

Sintaxe:

```
Search Include *fieldlist
Search Exclude *fieldlist
```

Você pode usar vários comandos Search para refinar sua seleção de campos para incluir. Os comandos são executados de cima para baixo.

Argumentos:

Argumentos

Argumento	Descrição
*fieldlist	Uma lista separada por vírgulas dos campos a serem incluídos ou excluídos na pesquisa inteligente. O uso de * como lista de campos indica todos os campos. Os caracteres curingas * e ? são permitidos nos nomes de campo. Poderá ser necessário colocar os nomes de campos entre aspas quando forem utilizados caracteres curinga.

Exemplo:

Exemplos de pesquisa

Comando	Descrição
`Search Include *;`	Inclua todos os campos na pesquisa inteligente.
`Search Exclude [*ID];`	Exclua todos os campos terminados em ID da pesquisa inteligente.
`Search Exclude '*ID';`	Exclua todos os campos terminados em ID da pesquisa inteligente.
`Search Include ProductID;`	Inclua o campo ProductID na pesquisa inteligente.

O resultado combinado desses três comandos, nesta sequência, é que todos os campos terminados em ID, exceto ProductID são excluídos da pesquisa inteligente.

Section

Com o comando **section**, é possível definir se os comandos **LOAD** e **SELECT** subsequentes devem ser considerados como dados ou como uma definição dos direitos de acesso.

Sintaxe:

```
Section (access | application)
```

Se nada for especificado, **section application** será assumido. A definição de **section** será válida até que um novo comando **section** seja criado.

Exemplo:

```
Section access;
Section application;
```

Select

A seleção de campos de uma fonte de dados ODBC ou provedor OLE DB é feita usando declarações SQL **SELECT** padrão. No entanto, a aceitação dos comandos **SELECT** dependem do driver ODBC ou provedor OLE DB utilizados. O uso da instrução **SELECT** requer uma conexão de dados aberta com a origem.

Sintaxe:

```
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 ]
```

Além disso, vários comandos **SELECT** podem ser concatenados por meio do uso de um operador **union**:

```
selectstatement Union selectstatement
```

O comando **SELECT** é interpretado pelo driver ODBC ou provedor OLE DB, portanto, os desvios da sintaxe geral de SQL podem ocorrer dependendo das capacidades dos drivers ODBC ou provedor OLE DB, por exemplo:

- **as** algumas vezes não é permitido, isto é, *aliasname* deve aparecer imediatamente após *fieldname*.
- **as** algumas vezes é obrigatório se *aliasname* for utilizado.
- **distinct**, **as**,**where**, **group by**, **order by** ou **union** às vezes não são compatíveis.
- Algumas vezes, o driver ODBC não aceita todos os diferentes sinais de aspas listados acima.

*Essa não é uma descrição completa do comando SQL **SELECT**! Por exemplo, os comandos **SELECT** podem ser aninhados, várias junções podem ser feitas em um comando **SELECT**, o número de funções permitidas nas expressões é às vezes muito grande etc.*

Argumentos:

Argumentos

Argumento	Descrição
distinct	**distinct** é um predicado utilizado caso as combinações de valores duplicadas nos campos selecionados devam ser carregadas somente uma vez.
distinctrow	**distinctrow** é um predicado utilizado caso os registros duplicados na tabela fonte devam ser carregados somente uma vez.
fieldlist	**fieldlist ::= (*| field) {, field }** Uma lista dos campos a serem selecionados. O uso de * como lista de campos indica todos os campos da tabela. **fieldlist ::= field {, field }** Uma lista de um ou mais campos, separados por vírgulas. **field ::= (fieldref | expression) [as aliasname]** A expressão pode por exemplo ser uma função numérica ou de caracteres (string) baseada em um ou vários outros campos. Alguns dos operadores e funções geralmente aceitos são: +, -, *, /, & (concatenação de cadeia de caracteres), sum(fieldname), count(fieldname), avg(fieldname)(average), month(fieldname), etc. Consulte a documentação do driver ODBC para obter mais informações. **fieldref ::= [tablename.] fieldname** As strings **tablename** e **fieldname** são strings de texto idênticas ao que implicam. Devem vir entre aspas duplas retas, se contiverem, por exemplo, espaços. A cláusula **as** é usada para atribuir um novo nome ao campo.
from	**tablelist ::= table {, table }** A lista de tabelas na qual os campos devem ser selecionados. **table ::= tablename [[as] aliasname]** O **tablename** pode ou não ser colocado entre aspas.
where	**where** é uma cláusula utilizada para declarar se um registro deve ou não ser incluído na seleção. **criterion** é uma expressão lógica que pode ser, algumas vezes, muito complexa. Alguns dos operadores aceitos são: operadores e funções numéricos, **=**, **<>** ou **#(**not equal), **>**, **>=**, **<**, **<=**, **and**, **or**, **not**, **exists**, **some**, **all**, **in** e também novos comandos **SELECT**. Consulte a documentação do driver ODBC ou provedor OLE DB para obter mais informações.
group by	**group by** é uma cláusula utilizada para agregar (agrupar) vários registros em um. Dentro de um grupo, para um determinado campo, todos os registros devem ter o mesmo valor ou o campo somente poderá ser usado a partir de uma expressão, como uma soma ou média. A expressão com base em um ou vários campos é definida na expressão do símbolo do campo.
having	**having** é uma cláusula usada para qualificar grupos de maneira semelhante à forma como a cláusula **where** é usada para qualificar registros.

Argumento	Descrição
order by	**order by** é uma cláusula utilizada para declarar a ordem de classificação da tabela resultante do comando **SELECT**.
join	**join** é um qualificador que define se várias tabelas devem ser combinadas em uma. Nomes de campos e de tabelas devem ser colocados entre aspas se contiverem espaços em branco ou letras dos conjuntos nacionais de caracteres. Quando o script é gerado automaticamente pelo Qlik Sense, as aspas utilizadas são as preferidas pelo driver ODBC ou provedor OLE DB especificados na definição da fonte de dados no comando **Connect**.

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

O comando **set** é usado para definir as variáveis do script. Essas variáveis podem ser utilizadas para substituir strings, caminhos, unidades e assim por diante.

Sintaxe:

```
Set variablename=string
```

Example 1:

```
Set FileToUse=Data1.csv;
```

Example 2:

```
Set Constant="My string";
```

Example 3:

```
Set BudgetYear=2012;
```

Sleep

O comando **sleep** pausa a execução do script pelo tempo especificado.

Sintaxe:

```
Sleep n
```

Argumentos:

Argumento	Descrição
n	Expressado em milissegundos, onde *n* é um número inteiro positivo, não superior a 3600000 (ou seja, 1 hora). O valor pode ser uma expressão.

Example 1:

```
sleep 10000;
```

Example 2:

```
sleep t*1000;
```

SQL

O comando **SQL** permite enviar um comando SQL arbitrário usando uma conexão ODBC ou OLE DB.

Sintaxe:

```
SQL sql_command
```

Enviar comandos SQL que atualizam o banco de dados retornará um erro se o Qlik Sense tiver aberto a conexão ODBC no modo apenas leitura.

A sintaxe:

```
SQL SELECT * from tab1;
```
é permitida, e é a sintaxe preferencial para **SELECT**, por motivos de coerência. O prefixo SQL, no entanto, permanecerá opcional para comandos **SELECT**.

Argumentos:

Argumento	Descrição
sql_command	Um comando SQL válido.

Example 1:

```
SQL leave;
```

Example 2:

```
SQL Execute <storedProc>;
```

SQLColumns

O comando **sqlcolumns** retorna um conjunto de campos que descreve as colunas de uma fonte de dados ODBC ouOLE DB com a qual é feita uma **connect**.

Sintaxe:

```
SQLcolumns
```

Os campos podem ser combinados com os campos gerados pelos comandos **sqltables** e **sqltypes** para fornecer uma boa visão geral de um banco de dados. Os doze campos padrão são:

TABLE_QUALIFIER

TABLE_OWNER

TABLE_NAME

COLUMN_NAME

DATA_TYPE

TYPE_NAME

PRECISION

LENGTH

SCALE

RADIX

NULLABLE

REMARKS

Para obter uma descrição detalhada desses campos, consulte um manual de referência ODBC.

Exemplo:

```
Connect to 'MS Access 7.0 Database; DBQ=C:\Course3\DataSrc\QWT.mbd';
SQLcolumns;
```

Alguns drivers ODBC podem não suportar esse comando. Alguns drivers ODBC podem produzir campos adicionais.

SQLTables

O comando **sqltables** retorna um conjunto de campos que descreve as tabelas de uma fonte de dados ODBC ouOLE DB com a qual é feita uma **connect**.

Sintaxe:

```
SQLTables
```

Os campos podem ser combinados com os campos gerados pelos comandos **sqlcolumns** e **sqltypes** para fornecer uma boa visão geral de um banco de dados. Os cinco campos padrão são:

TABLE_QUALIFIER

TABLE_OWNER

TABLE_NAME

TABLE_TYPE

REMARKS

Para obter uma descrição detalhada desses campos, consulte um manual de referência ODBC.

Exemplo:

```
Connect to 'MS Access 7.0 Database; DBQ=C:\Course3\DataSrc\QWT.mbd';
SQLTables;
```

Alguns drivers ODBC podem não suportar esse comando. Alguns drivers ODBC podem produzir campos adicionais.

SQLTypes

O comando **sqltypes** retorna um conjunto de campos que descreve os tipos de uma fonte de dados ODBC ou OLE DB com a qual é feita uma **connect**.

Sintaxe:

```
SQLTypes
```

Os campos podem ser combinados com os campos gerados pelos comandos **sqlcolumns** e **sqltables** para fornecer uma boa visão geral de um banco de dados. Os quinze campos padrão são:

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

Para obter uma descrição detalhada desses campos, consulte um manual de referência ODBC.

Exemplo:

```
Connect to 'MS Access 7.0 Database; DBQ=C:\Course3\DataSrc\QWT.mbd';
SQLTypes;
```

Alguns drivers ODBC podem não suportar esse comando. Alguns drivers ODBC podem produzir campos adicionais.

Star

É possível definir os caracteres utilizados para representar o conjunto de todos os valores de um campo na base de dados usando o comando **star**. Isso afeta os comandos **LOAD** e **SELECT** subsequentes.

Sintaxe:

```
Star is[ string ]
```

Argumentos:

Argumentos

Argumento	Descrição
string	Um texto arbitrário. Observe que a string deve estar entre sinais de aspas caso contenha espaços em branco. Se nada for especificado, **star is;** será adotado, isto é, não haverá nenhum símbolo star disponível a menos que seja especificado explicitamente. Essa definição será válida até que um novo comando **star** seja criada.

O comando **Star is** não é recomendado para uso na parte de dados do script (em **Aplicativo de seção**) se o section access for usado. No entanto, o caractere estrela é totalmente suportado para os campos protegidos na parte **Section Access** do script. Nesse caso, você não precisa usar o comando **Star is** explícito, pois isso está sempre implícito no section access.

Limitações

- Você não pode usar o caractere estrela com os campos-chave; isto é, campos que vinculam tabelas.
- Você não pode usar o caractere estrela com nenhum campo afetado pelo comando **Unqualify**, pois isso pode afetar os campos que vinculam tabelas.
- Você não pode usar o caractere estrela com tabelas não lógicas, por exemplo, tabelas de carregamento de informações ou tabelas de carregamento de mapeamento.
- Quando o caractere estrela é usado em um campo redutor (um campo vinculado aos dados) no section access, ele representa os valores listados nesse campo no section access. Ele não representa outros valores que possam existir nos dados, mas não estão listados no section access.
- Você não pode usar o caractere estrela com campos afetados por qualquer forma de redução de dados fora da área do **Section Access**.

Exemplo

O exemplo a seguir é um trecho de um script de carregamento de dados com seção de acesso.

```
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

];
```

O seguinte se aplica:

- O sinal *Star* é *.
- O usuário *ADMIN* vê todos os campos. Nada é omitido.
- O usuário *USER1* não pode ver o campo *SALES*.
- O usuário *USER2* não pode ver o campo *WAREHOUSE* .
- O usuário *USER3* não pode ver o campo *EMPLOYEES*.
- O usuário *USER4* é adicionado duas vezes à solução para OMITIR dois campos, *SALES* e *WAREHOUSE* para este usuário.
- O *USER5* tem um "*" adicionado, o que significa que todos os campos listados em OMIT estão indisponíveis, ou seja, o usuário *USER5* não consegue ver os campos *SALES*, *WAREHOUSE* e *EMPLOYEES* , mas pode ver o campo *ORDERS*.

Store

A instrução **Store** cria um arquivo QVD, Parquet, CSV ou TXT.

Sintaxe:

```
Store [ fieldlist from] table into filename [ format-spec ];
```

A declaração criará um arquivo de texto ou com nome QVD, Parquet explícito.

O comando só pode exportar campos de uma tabela lógica. Se os campos de várias tabelas forem exportados, uma join explícita deve ser feita previamente no script para criar a tabela de dados que deve ser exportada.

Os valores de texto são exportados para o arquivo CSV no formato UTF-8. É possível especificar um separador. Para isso, consulte **LOAD**. O comando **store** para um arquivo CSV não suporta exportação de BIFF.

Argumentos:

Armazenar argumentos de comando

Argumento	Descrição
*fieldlist::= (* | field) { , field })*	Uma lista dos campos a serem selecionados. O uso de * como lista de campos indica todos os campos. *field::= fieldname [***as** *aliasname]* *fieldname* é um texto idêntico a um nome de campo em *table*. (Observe que o nome do campo deverá estar entre aspas duplas ou colchetes se contiver espaços ou outros caracteres que não sejam padrão). *aliasname* é um nome alternativo do campo a ser usado no arquivo QVD ou CSV resultante.
table	Um rótulo de script representando uma tabela já carregada, que será usada como fonte dos dados.
filename	O nome do arquivo de destino, incluindo um caminho válido para uma conexão de dados da pasta existente. **Exemplo: *'lib://Table Files/target.qvd'*** No modo legado de uso de scripts, os seguintes formatos de caminho também são suportados: • absoluto **Exemplo: *c:\data\sales.qvd*** • relativo ao diretório operacional do Qlik Sense Qlik Cloud. **Exemplo: *data\sales.qvd*** Se o caminho for omitido, o Qlik Sense armazenará o arquivo no diretório especificado pelo comando **Directory**. Se não houver um comando **Directory**, o Qlik Sense armazenará o arquivo no diretório de trabalho, *C:\Users\{user}\Documents\Qlik\Sense\Apps*. •

Argumento	Descrição
*format-spec ::=***(** (**txt** | **qvd** | **parquet**), **compactação é** *codec***)**	Você pode definir a especificação do formato para qualquer um desses formatos de arquivo. Se a especificação do formato for omitida, **qvd** será assumido. • **txt** para arquivos CSV e TXT. • **qvd** para arquivos QVD. • **parquet** para arquivos Parquet. Se você usar **parquet**, também poderá definir qual codec de compactação usar com **compactação é**. Se você não especificar o codec de compactação com **compactação é**, snappy será usado. As seguintes configurações de compactação estão disponíveis: • uncompressed • snappy • gzip • lz4 • brotli • zstd • lz4_hadoop Exemplo: `Store mytable into [lib://DataFiles/myfile.parquet] (parquet, compression is lz4);`

Exemplos:

```
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';
```

A extensão do arquivo de conexões DataFiles faz distinção entre maiúsculas e minúsculas. Por exemplo: .qvd.

Table/Tables

As palavras chaves de script **Table** e **Tables** são usadas nos comandos **Drop**, **Comment** e **Rename**, bem como um especificador de formato em comandos **Load**.

Tag

Esse comando de script fornece uma maneira de atribuir tags a um ou mais campos ou tabelas. Se for feita uma tentativa de marcar um campo ou uma tabela não presente no aplicativo, a marcação será ignorada. Se houver ocorrências de um nome de campo ou tag em conflito, o último valor será usado.

Sintaxe:

```
Tag [field|fields] fieldlist with tagname
```

```
Tag [field|fields] fieldlist using mapname
```

```
Tag table tablelist with tagname
```

Argumentos

Argumento	Descrição
fieldlist	Um ou vários campos que devem ser marcados, em uma lista separada por vírgulas.
mapname	O nome de uma tabela de mapeamento carregada anteriormente em um comando **mapping Load** ou **mapping Select**.
tablelist	Uma lista separada por vírgulas das tabelas que devem ser marcadas.
tagname	Nome da tag que deve ser aplicada ao campo.

Example 1:

```
tagmap:
mapping LOAD * inline [
a,b
Alpha,MyTag
Num,MyTag
];
tag fields using tagmap;
```

Example 2:

```
tag field Alpha with 'MyTag2';
```

Trace

A declaração **trace** grava a string na janela **Progresso da execução do script** e no arquivo de log do script, quando utilizada. É muito útil para finalidades de depuração. Usando expansões $ de variáveis calculadas antes da declaração **trace**, você pode personalizar a mensagem.

Sintaxe:

```
Trace string
```

Example 1:

O comando a seguir pode ser usado logo após a instrução de Load que carrega a tabela "Main".

```
Trace Main table loaded;
```

Isso exibirá o texto "Main table loaded" no diálogo de execução do script e no arquivo de log.

Example 2:

Os comandos a seguir podem ser usados logo após a instrução de Load que carrega a tabela "Main".

```
Let MyMessage = NoOfRows('Main') & ' rows in Main table';

Trace $(MyMessage);
```

Isso exibirá um texto mostrando o número de linhas no diálogo de execução do script e no arquivo de log, por exemplo, "265,391 rows in Main table".

Unmap

O comando **Unmap** desativa o mapeamento do valor de campo especificado por um comando **Map ... Using** prévio de campos subsequentemente carregados.

Sintaxe:

```
Unmap *fieldlist
```

Argumentos:

Argumentos

Argumento	Descrição
*fieldlist	uma lista dos campos separados por vírgulas que não devem mais ser mapeados a partir desse ponto no script. O uso de * como lista de campos indica todos os campos. Os caracteres curingas * e ? são permitidos nos nomes de campo. Poderá ser necessário colocar os nomes de campos entre aspas quando forem utilizados caracteres curinga.

Exemplos e resultados:

Exemplo	Resultado
Unmap Country;	Desabilita o mapeamento do campo Country.
Unmap A, B, C;	Desabilita o mapeamento dos campos A, B e C.
Unmap * ;	Desabilita o mapeamento de todos os campos.

Unqualify

O comando **Unqualify** é utilizado para desativar a qualificação de nomes de campos previamente ativada pelo comando **Qualify**.

Sintaxe:

```
Unqualify *fieldlist
```

Argumentos:

Argumentos

Argumento	Descrição
*fieldlist	Uma lista separada por vírgula dos campos nos quais a qualificação deve ser ativada. O uso de * como lista de campos indica todos os campos. Os caracteres curingas * e ? são permitidos nos nomes de campo. Poderá ser necessário colocar os nomes de campos entre aspas quando forem utilizados caracteres curinga. Consulte a documentação sobre o comando **Qualify** para obter mais informações.

Example 1:

Em uma base de dados desconhecida, costuma ser útil, a princípio, assegurar-se de que apenas um ou poucos campos sejam associados, como ilustrado neste exemplo:

```
qualify *;
unqualify TransID;
SQL SELECT * from tab1;
SQL SELECT * from tab2;
SQL SELECT * from tab3;
```

Em primeiro lugar, a qualificação está ativada para todos os campos.
Em seguida, a qualificação é desativada para **TransID**.
Somente **TransID** será usado para associações entre as tabelas *tab1*, *tab2* e *tab3*. Todos os outros campos serão qualificados com o nome da tabela.

Untag

Esse comando de script fornece uma maneira de remover tags de campos ou tabelas. Se for feita uma tentativa de desmarcar um campo ou uma tabela não presente no aplicativo, a desmarcação será ignorada.

Sintaxe:

```
Untag [field|fields] fieldlist with tagname
```

```
Untag [field|fields] fieldlist using mapname
```

```
Untag table tablelist with tagname
```

Argumentos:

Argumentos

Argumento	Descrição
fieldlist	Um ou vários campos cujas tags devem ser removidas, em uma lista separada por vírgulas.
mapname	O nome de uma tabela de mapeamento carregada anteriormente em um comando de mapeamento **LOAD** ou **SELECT**.
tablelist	Uma lista separada por vírgulas das tabelas que devem ser desmarcadas.
tagname	Nome da tag que deve ser removida do campo.

Example 1:

```
tagmap:
mapping LOAD * inline [
a,b
Alpha,MyTag
Num,MyTag
];
Untag fields using tagmap;
```

Example 2:

```
Untag field Alpha with MyTag2;
```

2.6 Diretório de trabalho

Se você estiver fazendo referência a um arquivo em um comando de script e o caminho for omitido, o Qlik Sense pesquisa o arquivo na seguinte ordem:

1. O diretório especificado por um comando **Directory** (somente compatível em um modo de script legacy).
2. Se não houver um comando **Directory**, o Qlik Sense pesquisa no diretório de trabalho.

Diretório de trabalho do Qlik Sense Desktop

No Qlik Sense Desktop, o diretório de trabalho é *C:\Users\{user}\Documents\Qlik\Sense\Apps*.

Diretório de trabalho do Qlik Sense

Em uma instalação de servidor do Qlik Sense, o diretório de trabalho é especificado no Qlik Sense Repository Service; por padrão é *C:\ProgramData\Qlik\Sense\Apps*. Consulte a ajuda do Qlik Management Console para obter mais informações.

2 Trabalhando com variáveis no editor de carregamento de dados

Uma variável no Qlik Sense é um contêiner armazenando um valor estático ou um cálculo como, por exemplo, um valor numérico ou alfanumérico. Ao usar a variável no aplicativo, qualquer mudança feita será aplicada em qualquer lugar em que a variável for usada. Você pode definir variáveis na visão geral de variáveis ou no script, usando o Editor de carregamento de dados. Você define o valor de uma variável usando os comandos **Let** ou **Set** no script de carregamento de dados.

Você também pode trabalhar com as variáveis do Qlik Sense na visão geral de variáveis quando editar uma pasta.

2.7 Visão geral

Se o primeiro caractere do valor de uma variável for um sinal de igual ' = ', o Qlik Sense tentará avaliar o valor como uma fórmula (expressão Qlik Sense) e, em seguida, exibir ou retornar o resultado, em vez do texto real da fórmula.

Quando utilizada, a variável é substituída por seu valor. As variáveis podem ser usadas no script para a expansão de macros e em diversos comandos de controle. Isso é muito útil quando a mesma string é repetida várias vezes no script, por exemplo um caminho.

Algumas variáveis especiais do sistema serão definidas pelo Qlik Sense no início da execução do script, independentemente de seus valores anteriores.

2.8 Definindo uma variável

As variáveis fornecem a capacidade de armazenar valores estáticos ou o resultado de um cálculo. Ao definir uma variável, use a seguinte sintaxe:

```
set variablename = string
```

ou

```
let variable = expression
```

A instrução **Set** é usada para atribuição de strings. Ele atribui o texto à direita do sinal de igual à variável. A instrução **Let** avalia uma expressão à direita do sinal de igual no tempo de execução do script e atribui o resultado da expressão à variável.

As variáveis diferenciam maiúsculas de minúsculas.

No Qlik Sense, não é recomendado dar a uma variável o nome de um campo ou função.

Exemplos:

`set x = 3 + 4; //` a variável obterá a string '3 + 4' como o valor.

`let x = 3 + 4; //` retorna 7 como o valor.

`set x = Today(); //` retorna 'Today()' como o valor.

`let x = Today(); //` retorna a data de hoje como o valor, por exemplo, '27/9/2021'.

2.9 Excluindo uma variável

Se você remover uma variável do script e recarregar os dados, essa variável permanecerá no aplicativo. Se quiser remover por completo a variável do aplicativo, também deverá excluir a variável do diálogo de variáveis.

2.10 Carregando um valor de variável como um valor de campo

Se quiser carregar um valor de variável como um valor de campo em um comando **LOAD**, e o resultado da expansão de dólar for texto em vez de numérico ou uma expressão, você precisará colocar a variável expandida entre aspas simples.

Exemplo:

Este exemplo carrega a variável do sistema contendo a lista de erros de script para uma tabela. Você pode observar que a expansão do ScriptErrorCount na cláusula **If** não exige aspas, embora a expansão de ScriptErrorList exija.

```
IF $(ScriptErrorCount) >= 1 THEN

   LOAD '$(ScriptErrorList)' AS Error AutoGenerate 1;
END IF
```

2.11 Cálculo da variável

Existem várias maneiras de usar variáveis com valores calculados no Qlik Sense, e o resultado depende de como você as define e as chama em uma expressão.

Neste exemplo, carregamos alguns dados inline:

```
LOAD * INLINE [
    Dim, Sales
    A, 150
    A, 200
    B, 240
    B, 230
    C, 410
    C, 330
];
```

Vamos definir duas variáveis:

```
Let vSales  = 'Sum(Sales)' ;
Let vSales2  = '=Sum(Sales)' ;
```

Na segunda variável, adicionamos um sinal de igual antes da expressão. Assim, a variável será calculada antes de ser expandida e a expressão será avaliada.

Se você usar a variável vSales como está, por exemplo, em uma medida, o resultado será a string Sum(Sales), ou seja, nenhum cálculo será realizado.

Se você adicionar uma expansão de sinal de dólar e chamar $(vSales) na expressão, a variável será expandida e a soma de Sales, exibida.

Por fim, se você chamar $(vSales2), a variável será calculada antes de ser expandida. Isso significa que o resultado exibido é a soma total de Sales. A diferença entre usar =$(vSales) e =$(vSales2) como expressões de medida é observado neste gráfico que mostra os resultados:

Resultados

Dim	$(vSales)	$(vSales2)
A	350	1560
B	470	1560
C	740	1560

Como você pode ver, $(vSales) resulta na soma parcial para um valor de dimensão e $(vSales2) resulta na soma total.

Estão disponíveis as seguintes variáveis de script:

- *Variáveis de erro (page 274)*
- *Variáveis de interpretação numérica (page 209)*
- *Variáveis de sistema (page 201)*
- *Variáveis de tratamento de valores (page 207)*

2.12 Variáveis de sistema

As variáveisde sistema, algumas das quais são definidas pelo sistema, fornecem informações sobre o sistema e sobre o aplicativo Qlik Sense.

Visão geral das variáveis de sistema

Algumas das funções são descritas adicionalmente após a visão geral. Para essas funções específicas, você pode clicar no nome da função na sintaxe para acessar imediatamente seus detalhes.

CreateSearchIndexOnReload

Essa variável define se arquivos de índice de pesquisa devem ser criados durante a recarga dos dados.

```
CreateSearchIndexOnReload
```

Floppy
Retorna a letra da unidade do primeiro disco flexível encontrado, geralmente *a:*. Essa é uma variável definida pelo sistema.

```
Floppy
```

Esta variável não é suportada no modo padrão.

CD
Retorna a letra da primeira unidade de CD-ROM encontrada. Se nenhum CD-ROM for encontrado, *c:* será retornado. Essa é uma variável definida pelo sistema.

```
CD
```

Esta variável não é suportada no modo padrão.

HidePrefix
Todos os nomes de campos iniciados com essa string serão ocultos, da mesma forma que os campos de sistema. Essa é uma variável definida pelo usuário.

```
HidePrefix
```

HideSuffix
Todos os nomes de campos terminados com essa string serão ocultos, da mesma forma que os campos de sistema. Essa é uma variável definida pelo usuário.

```
HideSuffix
```

Include
A variável **Include/Must_Include** especifica um arquivo que contém texto, o qual deve ser incluído no script e avaliado como código de script. Ela não é usada para adicionar dados. Você pode armazenar partes do seu código de script em um arquivo de texto separado e reutilizá-lo em vários aplicativos. Essa é uma variável definida pelo usuário.

```
$(Include=filename)
$(Must_Include=filename)
```

OpenUrlTimeout
Esta variável define o timeout, em segundos, que o Qlik Sense deve respeitar quando obtiver dados de fontes URL (por exemplo, HTML páginas). Se a informação for omitida, o timeout será de 20 minutos.

```
OpenUrlTimeout
```

QvPath
Retorna os caracteres de busca do executável do Qlik Sense. Essa é uma variável definida pelo sistema.

```
QvPath
```

Esta variável não é suportada no modo padrão.

QvRoot
Retorna o diretório raiz do executável do Qlik Sense. Essa é uma variável definida pelo sistema.

```
QvRoot
```

Esta variável não é suportada no modo padrão.

QvWorkPath
Retorna os caracteres de busca do aplicativo atual do Qlik Sense. Essa é uma variável definida pelo sistema.

```
QvWorkPath
```

Esta variável não é suportada no modo padrão.

QvWorkRoot
Retorna o diretório raiz do aplicativo atual do Qlik Sense. Essa é uma variável definida pelo sistema.

```
QvWorkRoot
```

Esta variável não é suportada no modo padrão.

StripComments
Se essa variável estiver definida como 0, a remoção de comentários /*..*/ e // do script será impedida. Se essa variável não for definida, a remoção de comentários será sempre realizada.

```
StripComments
```

Verbatim
Em geral, são removidos automaticamente de todos os valores de campo os espaços em branco (ASCII 32) à esquerda ou à direita antes de serem carregados na base de dados do Qlik Sense . Se essa variável for definida como 1, a remoção de espaços em branco será suspensa. Os caracteres Tab (ASCII 9) e hard space (ANSI 160) nunca são removidos.

```
Verbatim
```

WinPath
Retorna os caracteres de busca para o Windows. Essa é uma variável definida pelo sistema.

```
WinPath
```

Esta variável não é suportada no modo padrão.

WinRoot
Retorna o diretório raiz do Windows. Essa é uma variável definida pelo sistema.

```
WinRoot
```

Esta variável não é suportada no modo padrão.

CollationLocale
Especifica qual localidade será usada para a ordem de classificação e correspondência da pesquisa. O valor é o nome de cultura de uma localidade, por exemplo, "en-US".Essa é uma variável definida pelo sistema.

```
CollationLocale
```

CreateSearchIndexOnReload

Essa variável define se arquivos de índice de pesquisa devem ser criados durante a recarga dos dados.

Sintaxe:

```
CreateSearchIndexOnReload
```

Você pode definir se os arquivos de índice de pesquisa devem ser criados durante a recarga dos dados ou após a primeira solicitação de pesquisa do usuário. A vantagem de criar arquivos de índice de pesquisa durante a recarga dos dados é que você evita o tempo de espera experimentado pelo primeiro usuário ao fazer uma pesquisa. Isso precisa ser ponderado contra o tempo de recarga de dados mais longo necessário pela criação do índice de pesquisa.

Se essa variável for omitida, os arquivos de índice de pesquisa não serão criados durante a recarga dos dados.

Para aplicativos de sessão, os arquivos de índice de pesquisa não serão criados durante a recarga dos dados, independentemente da configuração dessa variável.

Example 1: Criar arquivos de índice de pesquisa durante a recarga de dados

```
set CreateSearchIndexOnReload=1;
```

Example 2: Criar arquivos de índice de pesquisa após a primeira solicitação de pesquisa

```
set CreateSearchIndexOnReload=0;
```

HidePrefix

Todos os nomes de campos iniciados com essa string serão ocultos, da mesma forma que os campos de sistema. Essa é uma variável definida pelo usuário.

Sintaxe:

```
HidePrefix
```

Exemplo:

```
set HidePrefix='_' ;
```

Se esse comando for utilizado, os nomes de campo iniciados por um sublinhado não serão mostrados nas listas de nomes de campo quando os campos de sistema estiverem ocultos.

HideSuffix

Todos os nomes de campos terminados com essa string serão ocultos, da mesma forma que os campos de sistema. Essa é uma variável definida pelo usuário.

Sintaxe:

```
HideSuffix
```

Exemplo:

```
set HideSuffix='%';
```

Se esse comando for utilizado, os nomes de campo terminados em um sinal de porcentagem não serão mostrados nas listas de nomes de campo quando os campos de sistema estiverem ocultos.

Include

A variável **Include/Must_Include** especifica um arquivo que contém texto, o qual deve ser incluído no script e avaliado como código de script. Ela não é usada para adicionar dados. Você pode armazenar partes do seu código de script em um arquivo de texto separado e reutilizá-lo em vários aplicativos. Essa é uma variável definida pelo usuário.

Esta variável suporta apenas as conexões de dados da pasta no modo padrão.

Sintaxe:

```
$(Include=filename)
```

```
$(Must_Include=filename)
```

Existem duas versões da variável:

- **Include** não gera um erro se o arquivo não puder ser encontrado, ele falhará silenciosamente.
- **Must_Include** gera um erro se o arquivo não puder ser encontrado.

Se você não especificar um caminho, o nome do arquivo será relativo ao diretório de trabalho do aplicativo Qlik Sense. Você também pode especificar um caminho de arquivo absoluto ou um caminho para uma conexão de pasta lib://. Não coloque um caractere de espaço antes ou depois do sinal de igual.

*A construção **set Include** =filename não é aplicável.*

Exemplos:

```
$(Include=abc.txt);
```

```
$(Must_Include=lib://DataFiles/abc.txt);
```

Limitações

Compatibilidade cruzada limitada entre arquivos codificados em UTF-8 no Windows e no Linux.

O uso do UTF-8 com a BOM (Byte Order Mark, ou Marca de ordem de bytes) é opcional. A BOM pode interferir com o uso do UTF-8 em softwares que não esperam bytes não ASCII no início de um arquivo, mas que, de outra forma, poderiam lidar com o fluxo de texto.

- Os sistemas Windows usam a BOM em UTF-8 para identificar que um arquivo está codificado em UTF-8, mesmo não havendo ambiguidade no armazenamento de bytes.
- O Unix/Linux usa UTF-8 para Unicode, mas não usa a BOM, pois ela interfere na sintaxe de arquivos de comando.

Isso tem algumas implicações para o Qlik Sense.

- No Windows, qualquer arquivo que comece com uma BOM UTF-8 é considerado um arquivo de script UTF-8. Caso contrário, é assumida a codificação ANSI.
- No Linux, a página de código de 8 bits padrão do sistema é UTF-8. É por isso que o UTF-8 funciona, embora não contenha uma BOM.

Como resultado, a portabilidade não pode ser garantida. Nem sempre é possível criar um arquivo no Windows que possa ser interpretado pelo Linux, e vice-versa. Não há compatibilidade cruzada entre os dois sistemas em relação a arquivos codificados em UTF-8 devido ao tratamento diferente da BOM.

OpenUrlTimeout

Esta variável define o timeout, em segundos, que o Qlik Sense deve respeitar quando obtiver dados de fontes URL (por exemplo, HTML páginas). Se a informação for omitida, o timeout será de 20 minutos.

Sintaxe:

```
OpenUrlTimeout
```

Exemplo:

```
set OpenUrlTimeout=10;
```

StripComments

Se essa variável estiver definida como 0, a remoção de comentários /*..*/ e // do script será impedida. Se essa variável não for definida, a remoção de comentários será sempre realizada.

Sintaxe:

```
StripComments
```

Alguns drivers de banco de dados usam /*..*/ como dicas nos comandos **SELECT**. Nesse caso, os comentários não devem ser retirados antes de enviar o comando **SELECT** para o driver do banco de dados.

Recomenda-se que essa variável seja restaurada para 1 imediatamente após o(s) comando(s), onde for necessário.

Exemplo:

```
set StripComments=0;
SQL SELECT * /* <optimization directive> */ FROM Table ;
set StripComments=1;
```

Verbatim

Em geral, são removidos automaticamente de todos os valores de campo os espaços em branco (ASCII 32) à esquerda ou à direita antes de serem carregados na base de dados do Qlik Sense . Se essa variável for definida como 1, a remoção de espaços em branco será suspensa. Os caracteres Tab (ASCII 9) e hard space (ANSI 160) nunca são removidos.

Sintaxe:

```
Verbatim
```

Exemplo:

```
set Verbatim = 1;
```

2.13 Variáveis de tratamento de valores

Esta seção descreve as variáveis que são usadas para manipular valores NULL e outros.

Visão geral das variáveis de valores manipuláveis

Cada função é descrita adicionalmente após a visão geral. Você também pode clicar no nome da função na sintaxe para acessar imediatamente os detalhes dessa função específica.

NullDisplay
O símbolo definido substituirá todos os valores NULL do ODBC e conectores no nível mais baixo dos dados. Essa é uma variável definida pelo usuário.

```
NullDisplay
```

NullInterpret
Quando o símbolo definido ocorrer em um arquivo de texto, em um arquivo do Excel ou em um comando inline, ele será interpretado como NULL. Essa é uma variável definida pelo usuário.

```
NullInterpret
```

NullValue
Se a declaração **NullAsValue** for usada, o símbolo definido substituirá todos os valores NULL nos campos especificados como **NullAsValue** com a string especificada.

```
NullValue
```

OtherSymbol
Define um símbolo a ser tratado como “todos os outros valores” antes de um comando **LOAD**/**SELECT**. Essa é uma variável definida pelo usuário.

```
OtherSymbol
```

NullDisplay

O símbolo definido substituirá todos os valores NULL do ODBC e conectores no nível mais baixo dos dados. Essa é uma variável definida pelo usuário.

Sintaxe:

```
NullDisplay
```

Exemplo:

```
set NullDisplay='<NULL>';
```

NullInterpret

Quando o símbolo definido ocorrer em um arquivo de texto, em um arquivo do Excel ou em um comando inline, ele será interpretado como NULL. Essa é uma variável definida pelo usuário.

Sintaxe:

```
NullInterpret
```

Exemplos:

```
set NullInterpret=' ';
set NullInterpret =;
```

não retornará valores NULL para valores em branco no Excel, mas retornará para um arquivo de texto CSV

```
set NullInterpret ='';
```

retornará valores NULL para valores em branco no Excel.

NullValue

Se a declaração **NullAsValue** for usada, o símbolo definido substituirá todos os valores NULL nos campos especificados como **NullAsValue** com a string especificada.

Sintaxe:

```
NullValue
```

Exemplo:

```
NullAsValue Field1, Field2;
set NullValue='<NULL>';
```

OtherSymbol

Define um símbolo a ser tratado como “todos os outros valores” antes de um comando **LOAD/SELECT**. Essa é uma variável definida pelo usuário.

Sintaxe:

```
OtherSymbol
```

Exemplo:

```
set OtherSymbol='+';
LOAD * inline
[X, Y
a, a
b, b];
LOAD * inline
[X, Z
a, a
+, c];
```

O valor de campo Y='b' será agora vinculado a Z='c' através do outro símbolo.

2.14 Variáveis de interpretação numérica

Variáveis de interpretação numérica são definidas pelo sistema. Essas variáveis são incluídas na parte superior do script de carregamento e aplicam configurações de formatação numérica no momento da execução do script. Elas podem ser excluídas, editadas ou duplicadas.

Variáveis de interpretação numérica são geradas automaticamente de acordo com as configurações regionais atuais do sistema operacional quando um novo aplicativo é criado. No Qlik Sense Desktop, isso está de acordo com as configurações do sistema operacional do computador. No Qlik Sense, elas estão de acordo com o sistema operacional do servidor onde o Qlik Sense está instalado. Se o servidor do Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações

regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Formato da moeda

MoneyDecimalSep
O separador de decimal definido substitui o símbolo de decimal de moeda definido pelas configurações regionais.

```
MoneyDecimalSep
```

MoneyFormat
O símbolo definido substitui o símbolo de moeda definido pelas configurações regionais.

```
MoneyFormat
```

MoneyThousandSep
O separador de milhar definido substitui o símbolo de agrupamento de dígitos de moeda definido pelas configurações regionais.

```
MoneyThousandSep
```

Formato numérico

DecimalSep
O separador de decimal definido substitui o símbolo de decimal definido pelas configurações regionais.

```
DecimalSep
```

ThousandSep
O separador de milhar definido substitui o símbolo de agrupamento de dígitos do sistema operacional (configurações regionais).

```
ThousandSep
```

NumericalAbbreviation
A abreviatura numérica define qual abreviatura usar para os prefixos de escala de numerais, por exemplo, M para mega ou um milhão (10^6), e µ para micro (10^{-6}).

```
NumericalAbbreviation
```

Formato da hora

DateFormat
Essa variável de ambiente define o formato de data usado como padrão no aplicativo. O formato é usado para interpretar e formatar datas. Se a variável não estiver definida, o formato de data das configurações regionais do sistema operacional será usado quando o script for executado.

```
DateFormat
```

TimeFormat
O formato definido substitui o formato de hora do sistema operacional (configurações regionais).

```
TimeFormat
```

TimestampFormat
O formato definido substitui os formatos de datas e hora do sistema operacional (configurações regionais).

```
TimestampFormat
```

MonthNames
O formato definido substitui a convenção de nomes de meses das configurações regionais.

```
MonthNames
```

LongMonthNames
O formato definido substitui a convenção de nomes de meses longos nas configurações regionais.

```
LongMonthNames
```

DayNames
O formato definido substitui a convenção de nomes de dias da semana definida pelas configurações regionais.

```
DayNames
```

LongDayNames
O formato definido substitui a convenção de nomes de dias úteis longos nas configurações regionais.

```
LongDayNames
```

FirstWeekDay
Inteiro que define qual dia usar como o primeiro dia da semana.

```
FirstWeekDay
```

BrokenWeeks
Essa configuração define se as semanas são quebradas ou não.

```
BrokenWeeks
```

ReferenceDay
A configuração define qual dia em janeiro definir como dia de referência para definir a semana 1.

```
ReferenceDay
```

FirstMonthOfYear
A configuração define quais meses serão usados como o primeiro mês do ano, que pode ser usado para definir os anos fiscais que usam uma compensação mensal, por exemplo, com início em 1º de abril.

Esta configuração não está sendo usada no momento, mas está reservada para uso futuro.

As configurações válidas são 1º (de janeiro) a 12 (de dezembro). A configuração padrão é 1.

Sintaxe:

```
FirstMonthOfYear
```

Exemplo:

```
Set FirstMonthOfYear=4; //Sets the year to start in April
```

BrokenWeeks

Essa configuração define se as semanas são quebradas ou não.

Sintaxe:

```
BrokenWeeks
```

No Qlik Sense, as configurações regionais são obtidas quando o aplicativo é criado, e as configurações correspondentes são armazenadas no script como variáveis de ambiente.

Um desenvolvedor de aplicativos norte-americano geralmente obtém `Set BrokenWeeks=1;` no script, o que corresponde a semanas interrompidas. Um desenvolvedor de aplicativos europeu geralmente obtém `Set BrokenWeeks=0;` no script, correspondendo a semanas ininterruptas.

Semanas ininterruptas significam que:

- Em alguns anos, a semana 1 começa em dezembro e, em outros anos, a última semana do ano anterior continua em janeiro.
- De acordo com o ISO 8601, a semana 1 sempre tem pelo menos 4 dias em janeiro. No Qlik Sense, isso pode ser configurado usando a variável `ReferenceDay`.

Semanas quebradas significam que:

- A última semana do ano nunca continua em janeiro.
- A semana 1 começa em 1º de janeiro e, na maioria dos casos, não é uma semana completa.

Os seguintes valores podem ser usados:

- 0 (=usar semanas não quebradas)
- 1 (=usar semanas quebradas)

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas

para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplos:

Se quiser configurações ISO para semanas e números de semanas, certifique-se de ter o seguinte no script:

```
Set FirstWeekDay=0;
Set BrokenWeeks=0;   //(use unbroken weeks)
Set ReferenceDay=4;
```

Se quiser configurações dos EUA, certifique-se de ter o seguinte no script:

```
Set FirstWeekDay=6;
Set BrokenWeeks=1;   //(use broken weeks)
Set ReferenceDay=1;
```

DateFormat

Essa variável de ambiente define o formato de data usado como padrão no aplicativo e por data, retornando funções como `date()` e `date#()`. O formato é usado para interpretar e formatar datas. Se a variável não estiver definida, o formato de data definido pelas suas configurações regionais será obtido quando o script for executado.

Sintaxe:

```
DateFormat
```

Exemplos da função DateFormat

Exemplo	Resultado
`Set DateFormat='M/D/YY'; //(US format)`	Esse uso da função `DateFormat` define a data como o formato dos EUA: mês/dia/ano.
`Set DateFormat='DD/MM/YY'; //(UK date format)`	Esse uso da função `DateFormat` define a data como o formato do Reino Unido: dia/mês/ano.
`Set DateFormat='YYYY/MM/DD'; //(ISO date format)`	Esse uso da função `DateFormat` define a data como o formato ISO: ano/mês/dia.

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas

para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplo 1: Padrão de variáveis de sistema

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados de datas.
- A função `DateFormat`, que usará o formato de data dos EUA.

Neste exemplo, um conjunto de dados é carregado em uma tabela chamada "`Transactions`". Inclui um `campo` date. A definição de `DateFormat` dos EUA é usada. Esse padrão será usado para conversão implícita de texto em data quando as datas de texto forem carregadas.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `month`

Crie esta medida:

```
=sum(amount)
```

Tabela de resultados

date	mês	=sum(amount)
01/01/2022	Jan	1000
02/01/2022	Fev	2123
03/01/2022	Mar	4124
04/01/2022	Abr	2431

A definição de `DateFormat` MM/DD/AAAA é usada para conversão implícita de texto em datas, razão pela qual o campo `date` é interpretado corretamente como uma data. O mesmo formato é usado para exibir a data, conforme mostrado na tabela de resultados.

Exemplo 2: Alterar variável do sistema

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- O mesmo conjunto de dados do exemplo anterior.
- A função `DateFormat`, que usará o formato 'DD/MM/AAAA'.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `month`

Crie esta medida:

```
=sum(amount)
```

Tabela de resultados

date	mês	=sum(amount)
01/01/2022	Jan	1000
02/01/2022	Jan	2123
03/01/2022	Jan	4124
04/01/2022	Jan	2431

Como a definição `DateFormat` foi configurada como "DD/MM/AAAA", você pode ver que os dois dígitos após o primeiro símbolo "/" foram interpretados como o mês, fazendo com que todos os registros sejam do mês de janeiro.

Exemplo 3: Interpretação de datas

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados com datas em formato numérico.
- A variável `DateFormat`, que usará o formato 'DD/MM/AAAA'.
- A variável `date()`.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `month`

Crie esta medida:

```
=sum(amount)
```

Tabela de resultados

date	mês	=sum(amount)
06/03/2022	Jun	1000
06/04/2022	Jun	2123
06/05/2022	Jun	4124
06/07/2022	Jun	2431

No script de carregamento, você usa a função `date()` para converter a data numérica em um formato de data. Como você não fornece um formato especificado como um segundo argumento na função, `DateFormat` é usado. Isso resulta no campo de data usando o formato 'MM/DD/AAAA'.

Exemplo 4: Formatação de datas estrangeiras

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados de datas.
- A variável `DateFormat`, que usa o formato "DD/MM/AAAA", mas com barras de comentário excluídas.

Script de carregamento

```
// 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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `month`

Crie esta medida:

```
=sum(amount)
```

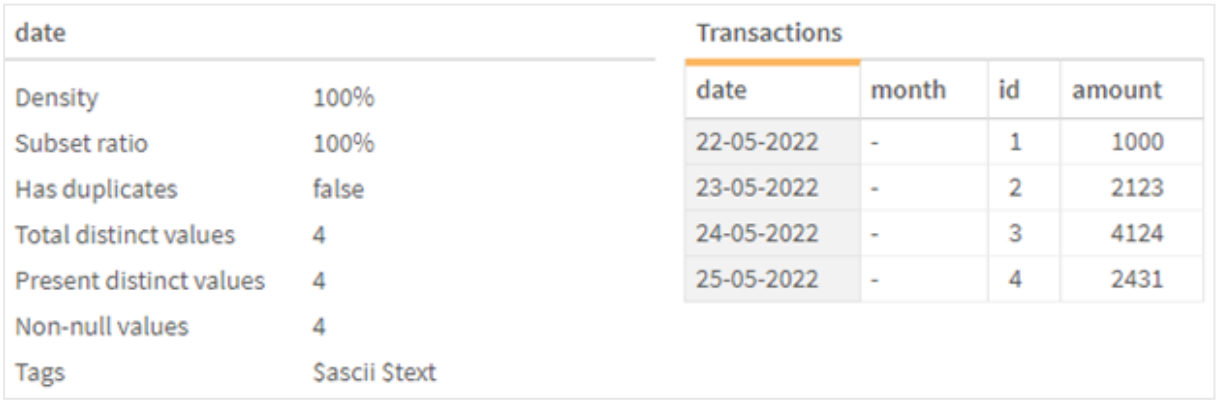
Tabela de resultados

date	mês	=sum(amount)
22-05-2022	-	1000
23-05-2022	-	2123
24-05-2022	-	4124
25-05-2022	-	2431

No script de carregamento inicial, o `DateFormat` que está sendo usado é o padrão 'MM/DD/AAAA'. Como o campo `date` no conjunto de dados de transações não está nesse formato, ele não é interpretado como uma data. Isso é mostrado na tabela de resultados em que os valores do campo `month` são nulos.

Você pode verificar os tipos de dados interpretados no Visualizador do modelo de dados inspecionando as `propriedades` "Tags" do campo date:

Visualização da tabela `Transactions`. Observe as "Tags" para o campo `date` indicando que os dados de entrada textual não foram implicitamente convertidos em um carimbo de data/hora.



Isso pode ser resolvido ativando a variável de sistema `DateFormat`:

```
// SET DateFormat='DD/MM/YYYY';
```

Remova as barras duplas e recarregue os dados.

Visualização da tabela `Transactions`. Observe as "Tags" para o campo `date` indicando que os dados de entrada textual foram implicitamente convertidos em um carimbo de data/hora.

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

O formato definido substitui a convenção de nomes de dias da semana definida pelas configurações regionais.

Sintaxe:

```
DayNames
```

Ao modificar a variável, é necessário um ponto e vírgula ; para separar os valores individuais.

Exemplos da função DayName

Exemplo de função	Definição do resultado
`Set DayNames='Mon;Tue;Wed;Thu;Fri;Sat;Sun';`	Esse uso da função DayNames define nomes de dias em seu formato abreviado.
`Set DayNames='M;Tu;W;Th;F;Sa;Su';`	Esse uso da função DayNames define nomes de dias pelas primeiras letras.

A função `DayNames` é frequentemente usada em combinação com as seguintes funções:

Funções relacionadas

Função	Interação
weekday (page 1076)	Função de script para retornar `DayNames` como valores de campos.
Date (page 1237)	Função de script para retornar `DayNames` como valores de campos.
LongDayNames (page 230)	Valores de formato longo de `DayNames`.

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações

regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplo 1: Padrão de variáveis de sistema

Script de carregamento e resultados

Visão geral

Neste exemplo, as datas no conjunto de dados são definidas no formato MM/DD/AAAA.

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados com datas, que será carregado em uma tabela denominada `Transactions`.
- Um campo `date`.
- A definição padrão de `DayNames`.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- date
- dayname

Crie esta medida:

```
sum(amount)
```

Tabela de resultados

date	dayname	sum(amount)
01/01/2022	Sáb	1000
02/01/2022	Ter	2123
03/01/2022	Ter	4124
04/01/2022	Sex	2431

No script de carregamento, a função WeekDay é usada com o campo date como o argumento fornecido. Na tabela de resultados, a saída dessa função WeekDay exibe os dias da semana no formato da definição DayNames.

Exemplo 2: Alterar variável de sistema

Script de carregamento e resultados

Visão geral

Abra o editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia. São usados o mesmo conjunto de dados e cenário do primeiro exemplo.

No entanto, no início do script, a definição DayNames é modificada para usar os dias abreviados da semana em africâner.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `dayname`

Crie esta medida:

```
sum(amount)
```

Tabela de resultados

date	dayname	sum(amount)
01/01/2022	Sa	1000
02/01/2022	Di	2123
03/01/2022	Di	4124
04/01/2022	Vr	2431

Na tabela de resultados, a saída dessa função `WeekDay` exibe os dias da semana no formato da definição `DayNames`.

É importante lembrar que, se o idioma de `DayNames` for modificado como neste exemplo, `LongDayNames` ainda conterá os dias da semana em inglês. Isso também precisaria ser modificado se ambas as variáveis fossem usadas no aplicativo.

Exemplo 3: Função de data

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados com datas, que será carregado em uma tabela denominada `Transactions`.
- Um campo `date`.
- A definição padrão de `DayNames`.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `dayname`

Crie esta medida:

```
sum(amount)
```

Tabela de resultados

date	dayname	sum(amount)
01/01/2022	Sáb	1000
02/01/2022	Ter	2123
03/01/2022	Ter	4124
04/01/2022	Sex	2431

A definição padrão de `DayNames` é usada. No script de carregamento, a função `Date` é usada com o campo `date` como o primeiro argumento. O segundo argumento é `WWW`. Essa formatação converte o resultado nos valores armazenados na definição `DayNames`. Isso é exibido na saída da tabela de resultados.

DecimalSep

O separador de decimal definido substitui o símbolo de decimal definido pelas configurações regionais.

O Qlik Sense interpreta texto automaticamente como números sempre que um padrão numérico reconhecível é encontrado. As `variáveis` de sistema `ThousandSep` e DecimalSep determinam a composição dos padrões aplicados ao analisar texto como números. As variáveis `ThousandSep` e `DecimalSep` definem o padrão de formato numérico ao visualizar conteúdo numérico em gráficos e tabelas front-end. Ou seja, ele afeta diretamente as opções de **Formato numérico** para qualquer expressão de front-end.

Assumindo um separador de milhar de vírgula "," e um separador decimal de ".", esses são exemplos de padrões que seriam implicitamente convertidos em valores numéricos equivalentes:

`0,000.00`

`0000.00`

`0,000`

Esses são exemplos de padrões que permaneceriam inalterados como texto, ou seja, não convertidos em padrões numéricos:

`0.000,00`

`0,00`

Sintaxe:

`DecimalSep`

Exemplos de funções

Exemplo	Resultado
`Set DecimalSep='.';`	Define "," como separador decimal.
`Set DecimalSep=',';`	Define "," como separador decimal.

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplo: efeito da configuração de variáveis separadoras de números em diferentes dados de entrada

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados de somas e datas, com as somas definidas em padrões de formato diferente.
- Uma tabela chamada `Transactions`.
- A variável `DecimalSep`, que está definida como `"."`.
- A variável `ThousandSep`, que está definida como `","`.
- A variável `delimiter`, que está definida como o caractere "|" para separar os diferentes campos em uma linha.

Script de carregamento

```
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 '|');
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão `amount`.

Crie esta medida:

```
=sum(amount)
```

Tabela de resultados

Amount	**=Sum(amount)**
Totais	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

Qualquer valor não interpretado como número permanece como texto e é alinhado à esquerda por padrão. Todos os valores convertidos com êxito são alinhados à direita, mantendo o formato de entrada original.

A coluna de expressão mostra o equivalente numérico, que, por padrão, é formatado apenas com um separador decimal ".". Isso pode ser substituído pela configuração da lista suspensa **Formato numérico** na configuração da expressão.

FirstWeekDay

Inteiro que define qual dia usar como o primeiro dia da semana.

Sintaxe:

```
FirstWeekDay
```

Segunda-feira é o primeiro dia da semana de acordo com a ISO 8601, o padrão internacional para representação de datas e horas. A segunda-feira também é usada como o primeiro dia da semana em vários países, por exemplo, no Reino Unido, França, Alemanha e Suécia.

Mas em outros países, como nos Estados Unidos e no Canadá, o domingo é considerado o início da semana.

No Qlik Sense, as configurações regionais são obtidas quando o aplicativo é criado, e as configurações correspondentes são armazenadas no script como variáveis de ambiente.

Um desenvolvedor de aplicativos norte-americano geralmente obtém `Set FirstWeekDay=6;` no script, o que corresponde ao domingo. Um desenvolvedor de aplicativos europeu geralmente obtém `Set FirstWeekDay=0;` no script, correspondendo à segunda-feira.

Valores que podem ser definidos para FirstWeekDay

Valor	Dia
0	Segunda-feira
1	Terça-feira
2	Quarta-feira
3	Quinta-feira
4	Sexta-feira
5	Sábado
6	Domingo

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações

regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplos:

Se quiser configurações ISO para semanas e números de semanas, certifique-se de ter o seguinte no script:

```
Set FirstWeekDay=0;  // Monday as first week day
Set BrokenWeeks=0;
Set ReferenceDay=4;
```

Se quiser configurações dos EUA, certifique-se de ter o seguinte no script:

```
Set FirstWeekDay=6;  // Sunday as first week day
Set BrokenWeeks=1;
Set ReferenceDay=1;
```

Exemplo 1 – Usando o valor padrão (script)

Script de carregamento e resultados

Visão geral

Abra o Editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia.

Neste exemplo, o script de carregamento usa o valor padrão da variável do sistema Qlik Sense, `FirstWeekDay=6`. Esses dados contêm dados dos primeiros 14 dias em 2020.

Script de carregamento

```
// 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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `week`
- `weekday`

Tabela de resultados

Date	**semana**	**weekday**
01/01/2021	1	Qua
01/02/2021	1	Qui
01/03/2021	1	Sex
01/04/2021	1	Sáb
01/05/2021	2	Dom
01/06/2020	2	Seg
01/07/2020	2	Ter
01/08/2020	2	Qua
01/09/2020	2	Qui
01/10/2020	2	Sex
01/11/2020	2	Sáb
01/12/2020	3	Dom
01/13/2020	3	Seg
01/14/2020	3	Ter

Como as configurações padrão estão sendo usadas, a variável do sistema `FirstWeekDay` é definida como 6. Na tabela de resultados, cada nova semana pode ser vista começando no domingo (5 e 12 de janeiro).

Exemplo 2 – Alterando a variável FirstWeekDay (script)

Script de carregamento e resultados

Visão geral

Abra o Editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia.

Neste exemplo, os dados contêm os primeiros 14 dias em 2020. No início do script, definimos a variável `FirstWeekDay` como 3.

Script de carregamento

```
// 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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `week`
- `weekday`

Tabela de resultados

Date	semana	weekday
01/01/2021	52	Qua
01/02/2021	1	Qui
01/03/2021	1	Sex
01/04/2021	1	Sáb
01/05/2021	1	Dom
01/06/2020	1	Seg
01/07/2020	1	Ter
01/08/2020	1	Qua
01/09/2020	2	Qui
01/10/2020	2	Sex
01/11/2020	2	Sáb
01/12/2020	2	Dom
01/13/2020	2	Seg
01/14/2020	2	Ter

Como a variável do sistema `FirstWeekDay` está definida como 3, o primeiro dia de cada semana será uma quinta-feira. Na tabela de resultados, cada nova semana pode ser vista a partir de quinta-feira (2 e 9 de janeiro).

LongDayNames

O formato definido substitui a convenção de nomes de dias úteis longos nas configurações regionais.

Sintaxe:

```
LongDayNames
```

O exemplo a seguir da função `LongDayNames` define os nomes dos dias na íntegra:

```
Set LongDayNames='Monday;Tuesday;Wednesday;Thursday;Friday;Saturday;Sunday';
```

Ao modificar a variável, é necessário um ponto e vírgula `;` para separar os valores individuais.

A função `LongDayNames` pode ser usada em combinação com a função *Date (page 1237)*, que retorna `DayNames` como valores de campo.

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações

regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplo 1: Padrão de variável de sistema

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados com datas, que será carregado em uma tabela denominada, `Transactions`.
- Um `campo` date.
- A definição padrão de `LongDayNames`.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `dayname`

Crie esta medida:

```
=sum(amount)
```

Tabela de resultados

date	dayname	=sum(amount)
01/01/2022	Sábado	1000
02/01/2022	Terça-feira	2123
03/01/2022	Terça-feira	4124
04/01/2022	Sexta-feira	2431

No script de carregamento, para criar um campo denominado `dayname`, a função `Date` é usada com o campo `date` como o primeiro argumento. O segundo argumento da função é a formatação `WWWW`.

O uso dessa formatação converte os valores do primeiro argumento no nome do dia inteiro correspondente definido na variável `LongDayNames`. Na tabela de resultados, os valores dos campos do nosso campo criado `dayname` exibem isso.

Exemplo 2: Alterar variável do sistema

Script de carregamento e resultados

Visão geral

Abra o editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia.

São usados o mesmo conjunto de dados e cenário do primeiro exemplo. No entanto, no início do script, a definição `LongDayNames` é modificada para usar os dias da semana em espanhol.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- date
- dayname

Crie esta medida:

```
=sum(amount)
```

Tabela de resultados

date	dayname	=sum(amount)
01/01/2022	Sábado	1000
02/01/2022	Martes	2123
03/01/2022	Martes	4124
04/01/2022	Viernes	2431

No script de carregamento, a variável LongDayNames é modificada para listar os dias da semana em espanhol.

Em seguida, você cria um campo denominado dayname, que é a função Date usada com o campo date como o primeiro argumento.

O segundo argumento da função é a formatação WWWW. Ao usar essa formatação, o Qlik Sense converte os valores do primeiro argumento no nome do dia inteiro correspondente definido na variável LongDayNames.

Na tabela de resultados, os valores de campos do nosso campo dayname criado exibem os dias da semana escritos em espanhol e na íntegra.

LongMonthNames

O formato definido substitui a convenção de nomes de meses longos nas configurações regionais.

Sintaxe:

LongMonthNames

Ao modificar a variável, é necessário usar ; para separar os valores individuais.

O exemplo a seguir da função LongMonthNames define os nomes dos meses na íntegra:

```
Set
LongMonthNames='January;February;March;April;May;June;July;August;September;October;November;D
ecember';
```

A função LongMonthNames é frequentemente usada em combinação com as seguintes funções:

Funções relacionadas

Função	Interação
Date (page 1237)	Função de script para retornar DayNamescomo valores de campos.
LongDayNames (page 230)	Valores de formato longo de DayNames.

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplo 1: Padrão de variáveis de sistema

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dadosde datas que é carregado em uma tabela denominada `Transactions`.
- Um campo `date`.
- A definição padrão de `LongMonthNames`.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões.

- `date`
- `monthname`

Crie esta medida

```
=sum(amount)
```

Tabela de resultados

date	monthname	sum(amount)
01/01/2022	Janeiro	1000.45
01/02/2022	Janeiro	2123.34
01/03/2022	Janeiro	4124.35
01/04/2022	Janeiro	2431.36
01/05/2022	Janeiro	4787.78
01/06/2022	Janeiro	2431.84
01/07/2022	Janeiro	2854.83
01/08/2022	Janeiro	3554.28
01/09/2022	Janeiro	3756.17
01/10/2022	Janeiro	3454.35

A definição padrão de `LongMonthNames` é usada. No script de carregamento, para criar um campo denominado `month`, a função `Date` é usada com o campo `date` como o primeiro argumento. O segundo argumento da função é a formatação `MMMM`.

Usando essa formatação, o Qlik Sense converte os valores do primeiro argumento no nome do mês inteiro correspondente definido na variável `LongMonthNames`. Na tabela de resultados, os valores dos campos do nosso campo criado `month` exibem isso.

Exemplo 2: Alterar variável de sistema

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados de datas que é carregado em uma tabela denominada `Transactions`.
- Um campo `date`.
- A variável `LongMonthNames`, que é modificada para usar os dias abreviados da semana em espanhol.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione `sum(amount)` como medida e estes campos como dimensões:

- `date`
- `monthname`

Crie esta medida:

```
=sum(amount)
```

Tabela de resultados

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

date	monthname	sum(amount)
01/07/2022	Enero	2854.83
01/08/2022	Enero	3554.28
01/09/2022	Enero	3756.17
01/10/2022	Enero	3454.35

No script de carregamento, a variável `LongMonthNames` é modificada para listar os meses do ano em espanhol. Em seguida, para criar um campo denominado `monthname`, a função `Date` é usada com o campo `date` como o primeiro argumento. O segundo argumento da função é a formatação `MMMM`.

Usando essa formatação, o Qlik Sense converte os valores do primeiro argumento no nome do mês inteiro correspondente definido na variável `LongMonthNames`. Na tabela de resultados, os valores do campo `monthname` criado exibem o nome do mês escrito em espanhol.

MoneyDecimalSep

O separador de decimal definido substitui o símbolo de decimal de moeda definido pelas configurações regionais.

Por padrão, o Qlik Sense exibe números e texto de forma diferente em gráficos de tabelas. Os números são alinhados à direita, e o texto é alinhado à esquerda. Isso facilita a localização de problemas de conversão de texto em número. Todas as tabelas nessa página que mostram resultados do Qlik Sense usarão essa formatação.

Sintaxe:

`MoneyDecimalSep`

Os aplicativos do Qlik Sense interpretarão os campos de texto que estão em conformidade com essa formatação como valores monetários. O campo de texto deve conter o símbolo de moeda definido na `variável` de sistema MoneyFormat. `MoneyDecimalSep` é particularmente útil ao lidar com fontes de dados recebidas de várias configurações regionais diferentes.

O exemplo a seguir mostra um possível uso da variável de sistema `MoneyDecimalSep`:

```
Set MoneyDecimalSep='.';
```

Essa função é frequentemente usada junto com as seguintes funções:

Funções relacionadas

Função	Interação
`MoneyFormat`	Em casos de interpretação de campos de texto, o símbolo `MoneyFormat` será usado como parte da interpretação. Para Formato numérico, a formatação `MoneyFormat` será usada pelo Qlik Sense em Objetos de gráfico.
`MoneyThousandSep`	Em casos de interpretação de campos de texto, a função `MoneyThousandSep` também deve ser seguida.

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplo 1: notação de ponto (.) para MoneyDecimalSep

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados que é carregado em uma tabela denominada `Transactions`.
- Dados fornecidos que têm seu campo monetário em formato de texto com um ponto “.” usado como separador de decimal. Cada registro também é prefixado por um símbolo “$”, exceto o último registro, que é prefixado por um símbolo “£”.

Lembre-se de que a variável de sistema `MoneyFormat` define o dólar “$” como a moeda padrão.

Script de carregamento

```
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'
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão:`amount`.

Adicione as seguintes medidas:

- `isNum(amount)`
- `sum(amount)`

Analise os resultados abaixo, demonstrando a interpretação correta somente de todos os valores em dólares "$".

Tabela de resultados

amount	=isNum(amount)	=Sum(amount)
Totais	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

A tabela de resultados acima mostra como o campo `amount` foi interpretado corretamente para todos os valores prefixados com dólar ($), enquanto o `amount` prefixado com libra (£) não foi convertido em um valor monetário.

Exemplo 2: notação de vírgula (,) para MoneyDecimalSep

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados que é carregado em uma tabela denominada `Transactions`.
- Dados fornecidos que têm seu campo monetário em formato de texto com uma vírgula “,” usada como separador decimal. Cada registro também é prefixado por um símbolo “$”, exceto o último registro, que usa erroneamente o separador decimal de ponto “.”.

Lembre-se de que a variável de sistema `MoneyFormat` define o dólar “$” como a moeda padrão.

Script de carregamento

```
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'
];
```

Resultados

Texto de parágrafo para resultados.

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão:`amount`.

Adicione as seguintes medidas:

- `isNum(amount)`
- `sum(amount)`

Revise os resultados abaixo, demonstrando a interpretação correta de todos os valores, exceto a quantia na qual o separador decimal usa a notação de ponto “.”' Nesse caso, uma vírgula deveria ter sido usada.

Tabela de resultados

amount	=isNum(amount)	=Sum(amount)
Totais	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

Essa variável de sistema define o padrão de formato usado pelo Qlik para a conversão automática de texto em número, onde o número é prefixado por um símbolo monetário. Também define como as medidas cujas propriedades de formato numérico estão definidas como “Dinheiro” serão exibidas em objetos de gráfico.

O símbolo definido como parte do padrão de formato na variável de sistema `MoneyFormat` substitui o símbolo monetário definido pelas configurações regionais.

Por padrão, o Qlik Sense exibe números e texto de forma diferente em gráficos de tabelas. Os números são alinhados à direita, e o texto é alinhado à esquerda. Isso facilita a localização de problemas de conversão de texto em número. Todas as tabelas nessa página que mostram resultados do Qlik Sense usarão essa formatação.

Sintaxe:

`MoneyFormat`

```
Set MoneyFormat='$ #,##0.00; ($ #,##0.00)';
```

Essa formatação será exibida em objetos de gráfico quando a propriedade Number Formatting de um `campo` numérico estiver definida como `Money`. Além disso, quando campos de texto numérico forem interpretados pelo Qlik Sense, se o símbolo monetário do campo de texto corresponder ao símbolo definido na variável `MoneyFormat`, o Qlik Sense interpretará esse campo como um valor monetário.

Essa função é frequentemente usada junto com as seguintes funções:

Funções relacionadas

Função	Interação
MoneyDecimalSep (page 237)	Para Formato numérico, `MoneyDecimalSep` será usado na formatação de campos de objetos.
MoneyThousandSep (page 245)	Para Formato numérico, `MoneyThousandSep` será usado na formatação de campos de objetos.

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplo 1: MoneyFormat

Script de carregamento e resultados

Visão geral

Abra o editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém um conjunto de dados que é carregado em uma tabela denominada `Transactions`. A definição padrão da variável `MoneyFormat` padrão é usada.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `amount`

Adicione esta medida:

```
=Sum(amount)
```

Em **Formato numérico**, selecione **Dinheiro** para configurar `Sum(amount)` como um valor monetário.

Tabela de resultados

date	Amount	=Sum(amount)
Totais		$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

A definição padrão de `MoneyFormat` é usada. Isso parece o seguinte: `$###0.00;-$###0.00`. Na tabela de resultados, o formato do campo `amount` exibe o símbolo da moeda, e o ponto decimal e as casas decimais foram incluídos.

Exemplo 2: MoneyFormat com separador de milhar e formatos de entrada mistos

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados de formato de entrada misto, que é carregado em uma tabela denominada `Transactions` com separadores de milhar e separadores decimais intercalados.
- Uma modificação da definição `MoneyFormat` é alterada para incluir uma vírgula como separador de milhar.
- Uma das linhas de dados delimitada erroneamente com vírgulas separadoras de milhar nos lugares errados. Observe como esse valor é deixado como texto e não pode ser interpretado como um número.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `amount`

Adicione esta medida:

```
=Sum(amount)
```

Em **Formato numérico**, selecione **Dinheiro** para configurar `Sum(amount)` como um valor monetário.

Tabela de resultados

date	Amount	=Sum(amount)
Totais		$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

No início do script, a variável de sistema `MoneyFormat` é modificada para incluir uma vírgula como separador de milhar. Na tabela Qlik Sense, pode-se ver que a formatação inclui esse separador. Além disso, a linha com o separador incorreto não foi interpretada corretamente e permanece como texto. É por isso que ela não contribui para a soma do valor.

MoneyThousandSep

O separador de milhar definido substitui o símbolo de agrupamento de dígitos de moeda definido pelas configurações regionais.

Por padrão, o Qlik Sense exibe números e texto de forma diferente em gráficos de tabelas. Os números são alinhados à direita, e o texto é alinhado à esquerda. Isso facilita a localização de problemas de conversão de texto em número. Todas as tabelas nessa página que mostram resultados do Qlik Sense usarão essa formatação.

Sintaxe:

```
MoneyThousandSep
```

Os aplicativos do Qlik Sense interpretarão os campos de texto que estão em conformidade com essa formatação como valores monetários. O campo de texto deve conter o símbolo de moeda definido na `variável` de sistema MoneyFormat. `MoneyThousandSep` é particularmente útil ao lidar com fontes de dados recebidas de várias configurações regionais diferentes.

O exemplo a seguir mostra um possível uso da variável de sistema `MoneyThousandSep`:

```
Set MoneyDecimalSep=',';
```

Essa função é frequentemente usada junto com as seguintes funções:

Funções relacionadas

Função	Interação
`MoneyFormat`	Em casos de interpretação de campos de texto, o símbolo `MoneyFormat` será usado como parte da interpretação. Para Formato numérico, a formatação `MoneyFormat` será usada pelo Qlik Sense em objetos de gráfico.
`MoneyDecimalSep`	Em casos de interpretação de campos de texto, a função `MoneyDecimalSep` também deve ser seguida.

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplo 1: notação de vírgula (,) para MoneyThousandSep

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados que é carregado em uma tabela denominada `Transactions`.
- Dados fornecidos que têm seu campo monetário em formato de texto com uma vírgula usada como separador de milhar. Cada registro também é prefixado por um símbolo “$”.

Lembre-se de que a variável de sistema `MoneyFormat` define o dólar “$” como a moeda padrão.

Script de carregamento

```
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'
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão:`amount`.

Adicione as seguintes medidas:

- `isNum(amount)`
- `sum(amount)`

Analise os resultados abaixo. A tabela demonstra a interpretação correta de todos os valores usando a notação de vírgula “,” como separador de milhar.

O campo `amount` foi interpretado corretamente para todos os valores, com exceção de um, que usou um ponto “.” como separador de milhar.

Tabela de resultados

amount	=isNum(amount)	=Sum(amount)
Totais	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

Exemplo 2: notação de ponto (.) para MoneyThousandSep

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados que é carregado em uma tabela denominada `Transactions`.
- Dados fornecidos que têm seu campo monetário em formato de texto com um ponto "." usado como separador de milhar. Cada registro também é prefixado por um símbolo "$".

Lembre-se de que a variável de sistema `MoneyFormat` define o dólar "$" como a moeda padrão.

Script de carregamento

```
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'
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão:`amount`.

Adicione as seguintes medidas:

- `isNum(amount)`
- `sum(amount)`

Analise os resultados abaixo, demonstrando a interpretação correta de todos os valores usando a notação de ponto “.” como separador de milhar.

O campo `amount` foi interpretado corretamente para todos os valores, com exceção de um, que usou uma vírgula “,” como separador de milhar.

Tabela de resultados

amount	=isNum(amount)	=Sum(amount)
Totais	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

O formato definido substitui a convenção de nomes de meses das configurações regionais.

Sintaxe:

MonthNames

Ao modificar a variável, é necessário usar ; para separar os valores individuais.

Exemplos de funções

Exemplo	Resultados
`Set MonthNames='Jan;Feb;Mar;Apr;May;Jun;Jul;Aug;Sep;Oct;Nov;Dec';`	Esse uso da função `MonthNames` define os nomes dos meses em inglês e em formato abreviado.
`Set MonthNames='Enero;Feb;Marzo;Abr;Mayo;Jun;Jul;Agosto;Set;Oct;Nov;Dic';`	Esse uso da função `MonthNames` define os nomes dos meses em espanhol e em formato abreviado.

A função `MonthNames` pode ser usada em combinação com as seguintes funções:

Funções relacionadas

Função	Interação
month (page 918)	Função de script para retornar valores definidos em `MonthNames` como valores de campos
Date (page 1237)	Função de script para retornar valores definidos em `MonthNames` como valores de campos com base em um argumento de formatação fornecido
LongMonthNames (page 233)	Valores de formato longo de `MonthNames`

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplo 1: Padrão de variáveis de sistema

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados de datas que é carregado em uma tabela denominada `Transactions`.
- Um campo `date`.
- A definição padrão de `MonthNames`.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `monthname`

Crie esta medida:

```
=sum(amount)
```

Tabela de resultados

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

A definição padrão de `MonthNames` é usada. No script de carregamento, a função `Month` é usada com o campo `date` como o argumento fornecido.

Na tabela de resultados, a saída dessa função `Month` exibe os meses do ano no formato da definição `MonthNames`.

Exemplo 2: Alterar variável de sistema

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados de datas que é carregado em uma tabela denominada `Transactions`.
- Um campo `date`.
- A variável `MonthNames` que é modificada para usar os meses abreviados em espanhol.

Script de carregamento

```
                          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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `monthname`

Crie esta medida:

```
=sum(amount)
```

Tabela de resultados

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

No script de carregamento, primeiro a variável `MonthNames` é modificada para listar os meses do ano abreviados em espanhol. A função `Month` é usada com o campo `date` como o argumento fornecido.

Na tabela de resultados, a saída dessa função `Month` exibe os meses do ano no formato da definição `MonthNames`.

É importante lembrar que, se o idioma da variável `MonthNames` for modificado como neste exemplo, a variável `LongMonthNames` ainda conterá os meses do ano em inglês. A variável `LongMonthNames` teria que ser modificada se ambas as variáveis fossem usadas no aplicativo.

Exemplo 3: Função de data

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados de datas que é carregado em uma tabela denominada `Transactions`.
- Um campo `date`.
- A definição padrão de `MonthNames`.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `monthname`

Crie esta medida:

```
=sum(amount)
```

Tabela de resultados

date	monthname	sum(amount)
01/01/2022	Jan	1000.45

date	monthname	sum(amount)
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

A definição padrão de `MonthNames` é usada. No script de carregamento, a função `Date` é usada com o campo `date` como o primeiro argumento. O segundo argumento é `MMM`.

Usando essa formatação, o Qlik Sense converte os valores do primeiro argumento no nome do mês correspondente definido na variável `MonthNames`. Na tabela de resultados, os valores dos campos do nosso campo criado `month` exibem isso.

NumericalAbbreviation

A abreviatura numérica define qual abreviatura usar para os prefixos de escala de numerais, por exemplo, M para mega ou um milhão (10^6), e µ para micro (10^{-6}).

Sintaxe:

```
NumericalAbbreviation
```

Você define a `variável` NumericalAbbreviation como uma cadeia de caracteres que contém uma lista de pares de definição de abreviação, delimitados por ponto e vírgula. Cada par de definições de abreviação deve conter a escala (o expoente na base decimal) e a abreviação separadas por dois pontos, por exemplo, `6:M` para um milhão.

A configuração padrão é `'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'`.

Exemplos:

Essa configuração mudará o prefixo de mil para t e o prefixo de um bilhão para B. Isso seria útil para aplicativos financeiros em que se espera abreviações como t$, M$ e B$.

```
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

A configuração define qual dia de janeiro definir como dia de referência para especificar a semana 1. Em outras palavras, essa configuração prescreve quantos dias na semana 1 devem ser datas dentro de janeiro.

Sintaxe:

```
ReferenceDay
```

`ReferenceDay` define quantos dias estão incluídos na primeira semana do ano. `ReferenceDay` pode ser definido como qualquer valor entre `1` e `7`. Qualquer valor fora da faixa de `1-7` é interpretado como o ponto médio da semana (`4`), o que equivale a definir `ReferenceDay` como `4`.

Se você não selecionar um valor para a configuração `ReferenceDay`, o valor padrão mostrará `ReferenceDay=0`, que será interpretado como o ponto médio da semana (`4`), como visto na tabela de valores de `ReferenceDay` abaixo.

A função `ReferenceDay` é frequentemente usada em combinação com as seguintes funções:

Funções relacionadas

Variável	Interação
BrokenWeeks (page 212)	Se o Qlik Sense Qlik Cloud estiver operando com semanas não quebradas, a configuração da `variável` ReferenceDay será aplicada. No entanto, se semanas não quebradas estiverem sendo usadas, a semana 1 começará em 1º de janeiro, terminará em conjunto com a configuração da variável `FirstWeekDay` e ignorará o sinalizador `ReferenceDay`.
FirstWeekDay (page 226)	Inteiro que define qual dia usar como o primeiro dia da semana.

O Qlik Sense permite que os seguintes valores sejam definidos para `ReferenceDay`:

Valores de ReferenceDay

Valor	Dia de referência
0 (padrão)	4 de janeiro
1	1º de janeiro
2	January 2
3	3 de janeiro
4	4 de janeiro
5	5 de janeiro
6	6 de janeiro
7	7 de janeiro

No exemplo a seguir, `ReferenceDay = 3` define 3 de janeiro como o dia de referência:

```
SET ReferenceDay=3; //(set January 3 as the reference day)
```

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplos:

Se quiser configurações ISO para semanas e números de semanas, certifique-se de ter o seguinte no script:

```
Set FirstWeekDay=0;
Set BrokenWeeks=0;
Set ReferenceDay=4;    // Jan 4th is always in week 1
```

Se quiser configurações dos EUA, certifique-se de ter o seguinte no script:

```
Set FirstWeekDay=6;
Set BrokenWeeks=1;
Set ReferenceDay=1;     // Jan 1st is always in week 1
```

Exemplo 1: Script de carregamento usando o valor padrão; ReferenceDay=0

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- A variável `ReferenceDay`, que está definida como `0`.
- A variável `BrokenWeeks` definida como `0` força o aplicativo a usar semanas não quebradas.
- Um conjunto de dados de datas do final de 2019 ao início de 2020.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `week`
- `weekday`

Tabela de resultados

date	**semana**	**weekday**
12/27/2019	52	Sex
12/28/2019	52	Sáb
12/29/2019	1	Dom
12/30/2019	1	Seg
12/31/2019	1	Ter
01/01/2020	1	Qua
01/02/2020	1	Qui
01/03/2020	1	Sex
01/04/2020	1	Sáb
01/05/2020	2	Dom
01/06/2020	2	Seg

date	semana	weekday
01/07/2020	2	Ter
01/08/2020	2	Qua
01/09/2020	2	Qui
01/10/2020	2	Sex
01/11/2020	2	Sáb

A semana 52 termina no sábado, 28 de dezembro. Como `ReferenceDay` exige que 4 de janeiro seja incluído na semana 1, essa começa em 29 de dezembro e termina no sábado, 4 de janeiro.

Exemplo: variável ReferenceDay definida como 5

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- A variável `ReferenceDay`, que está definida como `5`.
- A variável `BrokenWeeks` definida como `0` força o aplicativo a usar semanas não quebradas.
- Um conjunto de dados de datas do final de 2019 ao início de 2020.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `week`
- `weekday`

Tabela de resultados

date	semana	weekday
12/27/2019	52	Sex
12/28/2019	52	Sáb
12/29/2019	53	Dom
12/30/2019	53	Seg
12/31/2019	53	Ter
01/01/2020	53	Qua
01/02/2020	53	Qui
01/03/2020	53	Sex
01/04/2020	53	Sáb
01/05/2020	1	Dom
01/06/2020	1	Seg
01/07/2020	1	Ter
01/08/2020	1	Qua
01/09/2020	1	Qui
01/10/2020	1	Sex
01/11/2020	1	Sáb

A semana 52 termina no sábado, 28 de dezembro. A variável `BrokenWeeks` força o aplicativo a usar semanas não quebradas. O valor do dia de referência de `5` exige que 5 de janeiro seja incluído na semana 1.

No entanto, isso ocorre oito dias após a conclusão da semana 52 do ano anterior. Portanto, a semana 53 começa em 29 de dezembro e termina em 4 de janeiro. A semana 1 começa no domingo, 5 de janeiro.

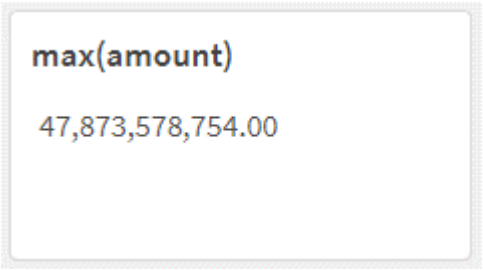
ThousandSep

O separador de milhar definido substitui o símbolo de agrupamento de dígitos do sistema operacional (configurações regionais).

Sintaxe:

ThousandSep

Objeto o Qlik Sense usando a variável ***ThousandSep*** *(com separador de milhar)*



Os aplicativos do Qlik Sense interpretam como números os campos de texto que estão em conformidade com essa formatação. Essa formatação será exibida em objetos de gráfico quando a propriedade **Formato numérico** do campo numérico estiver definida como **Número**.

ThousandSep é útil ao lidar com fontes de dados recebidas de várias configurações regionais.

> *Se a variável* ***ThousandSep*** *for modificada depois que objetos já tiverem sido criados e formatados no aplicativo, o usuário precisará reformatar cada campo relevante. Para isso, ele deve desmarcar e, em seguida, selecionar novamente a propriedade* ***Número*** *de* ***Formato numérico****.*

Os exemplos a seguir mostram os possíveis usos da variável de sistema 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)
```

Estes tópicos podem ajudar você a trabalhar com essa função:

Tópicos relacionados

Tópico	Descrição
DecimalSep (page 223)	Em casos de interpretação de campos de texto, as configurações do separador de decimal, conforme fornecidas por essa função, também devem ser respeitadas. Para Formato numérico, **DecimalSep** será usado pelo Qlik Sense quando necessário.

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução SET DateFormat no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplo 1: Variáveis padrão do sistema

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados que é carregado em uma tabela denominada `Transactions`.
- Uso da definição padrão da variável `ThousandSep`.

Script de carregamento

```
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
];
```

Resultados

Faça o seguinte:

1. Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão:`date`.
2. Adicione a seguinte medida:
   `=sum(amount)`

3. No painel de propriedades, em **Dados**, selecione a medida.
4. Em **Formato numérico**, selecione **Número**.

Ajustando o formato numérico para uma medida de gráfico

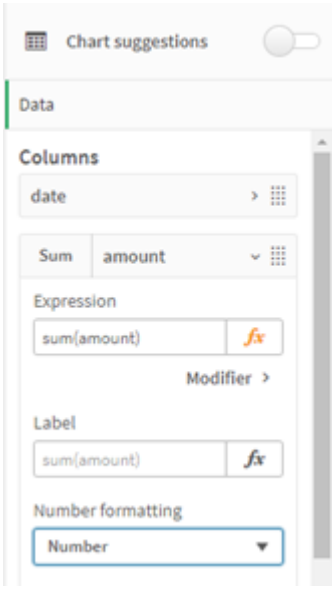


Tabela de resultados

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

Neste exemplo, é usada a definição padrão `ThousandSep`, especificada no formato de vírgula (“,”). Na tabela de resultados, o formato do campo de valor exibe uma vírgula entre agrupamentos de milhar.

Exemplo 2: Alterando a variável do sistema

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- O mesmo conjunto de dados do primeiro exemplo, que é carregado em uma tabela denominada `Transactions`.
- Modificação da definição `ThousandSep`, no início do script, para exibir um caractere "*" como separador de milhar. Esse é um exemplo extremo e é usado somente para demonstrar a funcionalidade da variável.

A modificação usada neste exemplo é extrema e não é comumente usada, mas é mostrada aqui para demonstrar a funcionalidade da variável.

Script de carregamento

```
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
];
```

Resultados

Faça o seguinte:

1. Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão:`date`.
2. Adicione a seguinte medida:

`=sum(amount)`

3. No painel de propriedades, em **Dados**, selecione a medida.
4. Em **Formato numérico**, selecione **Personalizado**.

Tabela de resultados

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

No início do script, a variável do sistema `ThousandSep` é modificada para "*". Na tabela de resultados, o formato do campo de valor pode ser visto para exibir um "*" entre o agrupamento de milhar.

Exemplo 3: Interpretação do texto

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados que é carregado em uma tabela denominada `Transactions`.
- Dados que têm seu campo numérico em formato de texto, com uma vírgula usada como separador de milhar.
- Uso da variável de sistema `ThousandSep` padrão.

Script de carregamento

```
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'
];
```

Resultados

Faça o seguinte:

1. Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão:`date`.
2. Adicione a seguinte medida:
   `=sum(amount)`
3. No painel de propriedades, em **Dados**, selecione a medida.
4. Em **Formato numérico**, selecione **Número**.
5. Adicione a medida a seguir para avaliar se o campo de valor é ou não um valor numérico:
   `=isnum(amount)`

Tabela de resultados

date	=sum(amount)	=isnum(amount)
01/01/2022	10,000,000,441.00	-1
01/02/2022	21,492,432.00	-1
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

Depois que os dados são carregados, podemos ver que o Qlik Sense interpretou o campo de quantidade como um valor numérico pelo fato de os dados estarem em conformidade com a variável `ThousandSep`. Isso é demonstrado pela função `isnum()`, que avalia cada entrada como -1, ou `TRUE`.

No Qlik Sense, o valor booleano “true” é representado por -1, e o valor falso é representado por 0.

TimeFormat

O formato definido substitui o formato de hora do sistema operacional (configurações regionais).

Sintaxe:

```
TimeFormat
```

Exemplo:

```
Set TimeFormat='hh:mm:ss';
```

TimestampFormat

O formato definido substitui os formatos de datas e hora do sistema operacional (configurações regionais).

Sintaxe:

```
TimestampFormat
```

Exemplo:

Os exemplos a seguir usam *1983-12-14T13:15:30Z* como dados de carimbo de data/hora para mostrar os resultados de diferentes instruções **SET TimestampFormat**. O formato de data usado é **YYYYMMDD** e o formato de hora é **h:mm:ss TT**. O formato de data é especificado no comando **SET DateFormat** e o formato de hora é especificado no comando **SET TimeFormat**, na parte superior do script de carregamento de dados.

Resultados

Exemplo	Resultado
`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

Exemplos: Script de carregamento

Exemplo: Script de carregamento
No primeiro script de carregamento, *SET TimestampFormat='DD/MM/YYYY h:mm:ss[.fff] TT'* é usado. No segundo script de carregamento, o formato do carimbo de data/hora é alterado para *SET TimestampFormat='MM/DD/YYYY hh:mm:ss[.fff]'*. Os diferentes resultados mostram como o comando **SET TimeFormat** funciona com diferentes formatos de dados de hora.

A tabela abaixo mostra o conjunto de dados usado nos scripts de carregamento a seguir. A segunda coluna da tabela mostra o formato de cada registro de data e hora no conjunto de dados. Os cinco primeiros carimbos de data/hora seguem regras ISO 8601, mas o sexto não.

Conjunto de dados

Tabela mostrando os dados de hora usados e o formato de cada carimbo de data/hora no conjunto de dados.

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

No **Editor de carregamento de dados**, crie uma nova seção e, em seguida, adicione o script de exemplo e execute-o. Em seguida, adicione pelo menos os campos listados na coluna de resultados a uma pasta em seu aplicativo para ver o resultado.

Script de carregamento

```
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
];
```

Resultados

Tabela Qlik Sense mostrando resultados da variável de interpretação TimestampFormat em uso no script de carregamento. O último carimbo de data/hora no conjunto de dados não retorna uma data correta.

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	-

O próximo script de carregamento usa o mesmo conjunto de dados. No entanto, ele usa *SET TimestampFormat='MM/DD/YYYY hh:mm:ss[.fff]'* para corresponder o formato não ISO 8601 do sexto carimbo de data/hora.

No **Editor de carregamento de dados**, substitua o script de exemplo anterior pelo script abaixo e execute-o. Em seguida, adicione pelo menos os campos listados na coluna de resultados a uma pasta para ver o resultado.

Script de carregamento

```
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
];
```

Resultados

Tabela Qlik Sense mostrando resultados da variável de interpretação TimestampFormat em uso no script de carregamento.

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

2.15 Variáveis do Direct Discovery

Variáveis de sistema do Direct Discovery

DirectCacheSeconds
É possível definir um limite de armazenamento em cache para os resultados da consulta do Direct Discovery para visualizações. Depois que este limite for alcançado, o Qlik Sense limpa o cache quando novas consultas do Direct Discovery forem feitas. O Qlik Sense consulta as seleções nos dados de origem e cria o cache novamente com base no limite de tempo designado. O resultado para cada combinação de seleções é armazenado em cache de forma independente. Ou seja, o cache é atualizado para cada seleção de forma independente; portanto, uma seleção atualiza o cache apenas para os campos selecionados e uma segunda seleção atualiza o cache para os seus campos relevantes. Se a segunda seleção incluir campos que foram atualizados na primeira seleção, eles não serão atualizados no cache novamente se o limite de armazenamento em cache não tiver sido atingido.

O cache do Direct Discovery não se aplica às visualizações de **Tabela**. As seleções de tabela sempre consultam a origem dos dados.

O valor limite deve ser definido em segundos. O limite padrão do cache é de 1.800 segundos (30 minutos).

O valor usado para **DirectCacheSeconds** é o valor definido no momento em que o comando **DIRECT QUERY** é executado. O valor não pode ser alterado no tempo de execução.

Exemplo:

```
SET DirectCacheSeconds=1800;
```

DirectConnectionMax
É possível fazer chamadas assíncronas paralelas para o banco de dados usando o recurso de pool de conexão. A sintaxe do script de carga para configurar o recurso de pool é a seguinte:

```
SET DirectConnectionMax=10;
```

A configuração numérica especifica o número máximo de conexões de banco de dados que o código do Direct Discovery deve usar ao atualizar uma pasta. A configuração padrão é 1.

Esta variável deve ser usada com cautela. Defini-la para maior que 1 é conhecido por causar problemas ao se conectar ao Microsoft SQL Server.

DirectUnicodeStrings
O Direct Discovery pode suportar a seleção de dados Unicode estendidos usando o formato padrão de SQL para literais dos caracteres estendidos (N'<extended string>'), conforme exigido por alguns bancos de dados (especialmente o SQL Server). O uso dessa sintaxe pode ser ativado para o Direct Discovery com a variável de script **DirectUnicodeStrings**.

Definir essa variável para "true" habilitará o uso do marcador de caracteres ANSI generalizado e padrão "N" na frente das literais dos caracteres. Nem todos os bancos de dados suportam este padrão. A configuração padrão é "false".

DirectDistinctSupport
Quando um valor do campo **DIMENSION** é selecionado em um objeto do Qlik Sense, uma consulta é gerada para o banco de dados de origem. Quando a consulta exigir o agrupamento, o Direct Discovery usa a palavra-chave **DISTINCT** para selecionar apenas valores exclusivos. No entanto, alguns bancos de dados exigem a palavra-chave **GROUP BY**. Defina **DirectDistinctSupport** como 'false' para gerar **GROUP BY** em vez de **DISTINCT** em consultas para obter valores exclusivos.

```
SET DirectDistinctSupport='false';
```
Se DirectDistinctSupport for definido para "true", **DISTINCT**será usado. Caso contrário, o comportamento padrão é usar **DISTINCT**.

DirectEnableSubquery
Em cenários multitabela de alta cardinalidade, é possível gerar subconsultas na consulta SQL em vez de gerar uma cláusula IN grande. Essa opção é ativada ao configurar **DirectEnableSubquery** como 'true'. O valor padrão é 'false'.

*Quando **DirectEnableSubquery** for desabilitado, não será possível carregar tabelas que não estiverem no modo Direct Discovery.*

```
SET DirectEnableSubquery='true';
```

Variáveis de marcação da consulta do Teradata

A marcação da consulta do Teradata é uma função que permite a colaboração dos aplicativos corporativos com o banco de dados subjacente do Teradata, a fim de fornecer uma melhor contabilidade, priorização e gerenciamento de carga de trabalho. Ao usar a marcação da consulta, é possível retornar metadados, como as credenciais do usuário, para uma consulta.

Duas variáveis estão disponíveis, que são caracteres que são avaliados e enviados para o banco de dados.

SQLSessionPrefix

Estes caracteres são enviados quando uma conexão com o banco de dados é criada.

```
SET SQLSessionPrefix = 'SET QUERY_BAND = ' & Chr(39) & 'Who=' & OSuser() & ';' & Chr(39) & '
FOR SESSION;';
```

Se **OSuser()**, por exemplo, retornar *WA\sbt*, isso será avaliado para `SET QUERY_BAND = 'Who=WA\sbt;' FOR SESSION;`, que é enviado para o banco de dados quando a conexão é criada.

SQLQueryPrefix

Estes caracteres são enviados para cada consulta individual.

```
SET SQLSessionPrefix = 'SET QUERY_BAND = ' & Chr(39) & 'Who=' & OSuser() & ';' & Chr(39) & '
FOR TRANSACTION;';
```

Variáveisde caracteres do Direct Discovery

DirectFieldColumnDelimiter

É possível definir o caractere usado como delimitador de campo nos comandos **Direct Query** para os bancos de dados exigem um caractere diferente de vírgula como o delimitador de campo. O caractere especificado deve estar entre aspas simples no comando **SET**.

```
SET DirectFieldColumnDelimiter= '|'
```

DirectStringQuoteChar

É possível especificar que um caractere use caracteres entre aspas em uma consulta gerada. O padrão são aspas simples. O caractere especificado deve estar entre aspas simples no comando **SET**.

```
SET DirectStringQuoteChar= '"';
```

DirectIdentifierQuoteStyle

É possível especificar que aspas de identificadores diferentes de ANSI sejam usadas nas consultas geradas. Desta vez, as únicas aspas diferentes deANSI disponíveis é GoogleBQ. O padrão é ANSI. Maiúsculas, minúsculas e letras maiúsculas e minúsculas podem ser usadas (ANSI, ansi, Ansi).

```
SET DirectIdentifierQuoteStyle="GoogleBQ";
```

Por exemplo, as aspas do ANSI são usadas no seguinte comando **SELECT**:

```
SELECT [Quarter] FROM [qvTest].[sales] GROUP BY [Quarter]
```

Quando **DirectIdentifierQuoteStyle** é definido como "GoogleBQ", o comando **SELECT** usaria as aspas da seguinte forma:

```
SELECT [Quarter] FROM [qvTest.sales] GROUP BY [Quarter]
```

DirectIdentifierQuoteChar

É possível especificar que um caractere controle as aspas de identificadores em uma consulta gerada. Isso pode ser definido para um caractere (como aspas duplas) ou dois (como um par de colchetes). O padrão são aspas duplas.

```
SET DirectIdentifierQuoteChar='[]';
SET DirectIdentifierQuoteChar='``';
SET DirectIdentifierQuoteChar=' ';
SET DirectIdentifierQuoteChar='""';
```

DirectTableBoxListThreshold
Quando os campos do Direct Discovery são usados em uma visualização de **Tabelas**, um limite é definido para limitar o número de linhas exibidas. O limite padrão é de 1.000 registros. A configuração do limite padrão pode ser alterada ao definir a variável**DirectTableBoxListThreshold** no script de carregamento. Por exemplo:

```
SET DirectTableBoxListThreshold=5000;
```
A configuração do limite aplica-se apenas às visualizações de **Tabelas** que contenham campos do Direct Discovery. Visualizações de **tabela** que contém apenas campos com memória não estão limitados pela configuração **DirectTableBoxListThreshold**.

Nenhum campo é exibido na visualização de **Tabelas**, até que a seleção tenha menos registros do que o limite.

Variáveis de interpretação numérica do Direct Discovery

DirectMoneyDecimalSep
O separador de decimais definido substitui o símbolo decimal da moeda no comando do SQL gerado para carregar dados com o Direct Discovery. Este caractere deve corresponder ao caractere utilizado em **DirectMoneyFormat**.

O valor padrão é `'.'`

Exemplo:

```
Set DirectMoneyDecimalSep='.';
```

DirectMoneyFormat
O símbolo definido substitui o formato de moeda no comando do SQL gerado para carregar os dados com o Direct Discovery. O símbolo de moeda para o separador de milhares não deve ser incluído.

O valor padrão é `'#.0000'`

Exemplo:

```
Set DirectMoneyFormat='#.0000';
```

DirectTimeFormat
O formato de tempo definido substitui o formato de tempo no comando do SQL gerado para carregar os dados com o Direct Discovery.

Exemplo:

```
Set DirectTimeFormat='hh:mm:ss';
```

DirectDateFormat
O formato de data definido substitui o formato de data no comando do SQL gerado para carregar os dados com o Direct Discovery.

Exemplo:

```
Set DirectDateFormat='MM/DD/YYYY';
```

DirectTimeStampFormat
O formato definido substitui o formato de data e tempo no comando do SQL gerado no comando do SQL gerado para carregar os dados com o Direct Discovery.

Exemplo:

```
Set DirectTimestampFormat='M/D/YY hh:mm:ss[.fff]';
```

2.16 Variáveis de erro

Os valores de todas as variáveis de erro existirão após a execução do script. A primeira variável, ErrorMode, é a entrada do usuário, e as três últimas são saídas do Qlik Sense com informações sobre os erros no script.

Visão geral das variáveis de erro

Cada variável é descrita mais adiante após a visão geral. Você também pode clicar no nome da variável na sintaxe para acessar imediatamente os detalhes dessa variável específica.

Consulte a ajuda online do Qlik Sense para obter mais detalhes sobre as variáveis.

ErrorMode
Essa variável de erro determina a ação a ser executada pelo Qlik Sense quando um erro for encontrado durante a execução do script.

```
ErrorMode
```

ScriptError
Essa variável de erro retorna o código de erro do último comando de script executado.

```
ScriptError
```

ScriptErrorCount
Essa variável de erro retorna o número total de declarações que causaram erros durante a execução do script atual. Esta variável é sempre restaurada para 0 no início da execução do script.

```
ScriptErrorCount
```

ScriptErrorList
Essa variável de erro conterá uma lista concatenada de todos os erros de script ocorridos durante a última execução do script. Cada erro é separado por uma linha.

```
ScriptErrorList
```

ErrorMode

Essa variável de erro determina a ação a ser executada pelo Qlik Sense quando um erro for encontrado durante a execução do script.

Sintaxe:

```
ErrorMode
```

Argumentos:

Argumentos

Argumento	Descrição
ErrorMode=1	A configuração padrão. A execução do script será pausada e o usuário será solicitado a executar uma ação (modo não batch).
ErrorMode =0	O Qlik Sense simplesmente ignora a falha e continua a execução do script no próximo comando de script.
ErrorMode =2	O Qlik Sense disparará uma mensagem de erro “Falha na execução do script…” imediatamente em caso de falha, sem solicitar antecipadamente que o usuário execute uma ação.

Exemplo:

```
set ErrorMode=0;
```

ScriptError

Essa variável de erro retorna o código de erro do último comando de script executado.

Sintaxe:

```
ScriptError
```

Esta variável será restaurada para 0 após cada execução bem-sucedida do comando de script. Em caso de erro, será definida como um código de erro interno do Qlik Sense. Os códigos de erro são valores duais que incluem um componente numérico e outro de texto. Códigos de erro existentes:

Códigos de erro de script

Código de erro	Descrição
0	Sem erro. O texto de valor duplo está vazio.
1	Erro geral.
2	Erro de sintaxe.
3	Erro geral de ODBC.
4	Erro geral de OLE DB.
5	Erro de banco de dados personalizado geral.
6	Erro geral de XML.
7	Erro geral de HTML.

Código de erro	Descrição
8	Arquivo não encontrado.
9	Banco de dados não encontrado.
10	Tabela não encontrada.
11	Campo não encontrado.
12	Arquivo com formato incorreto.
16	Erro semântico.

Exemplo:

```
set ErrorMode=0;

LOAD * from abc.qvf;

if ScriptError=8 then

exit script;

//no file;

end if
```

ScriptErrorCount

Essa variável de erro retorna o número total de declarações que causaram erros durante a execução do script atual. Esta variável é sempre restaurada para 0 no início da execução do script.

Sintaxe:

```
ScriptErrorCount
```

ScriptErrorList

Essa variável de erro conterá uma lista concatenada de todos os erros de script ocorridos durante a última execução do script. Cada erro é separado por uma linha.

Sintaxe:

```
ScriptErrorList
```

2 Expressões de script

As expressões podem ser usadas nos comandos **LOAD** e **SELECT**. A sintaxe e as funções descritas aqui aplicam-se ao comando **LOAD** e não ao comando **SELECT**, pois o último é interpretado pelo driver ODBC e não pelo Qlik Sense. No entanto, a maioria dos drivers ODBC geralmente é capaz de interpretar várias funções descritas a seguir.

As expressões consistem em funções, campos e operadores, combinados em uma sintaxe.

Todas as expressões em um script do Qlik Sense retornam um número e/ou uma string – o que for adequado. As funções e operadores lógicos retornam 0 False e -1 para True. As conversões de número para caractere e vice-versa estão implícitas. As funções e operadores lógicos interpretam 0 como False e tudo o mais como True.

A sintaxe geral de uma expressão é:

Sintaxe geral

Expressão	**Campos**	**Operador**
expression ::= (constant	constant	|
expression ::= (constant	fieldref	|
expression ::= (constant	operator1 expression	|
expression ::= (constant	expression operator2 expression	|
expression ::= (constant	function	|
expression ::= (constant	(expression)	)

na qual:

- **constant** é uma string (um texto, data ou hora) entre aspas simples retas ou um número. Constantes são escritas sem separador de milhar e com um ponto decimal como separador de decimal.
- **fieldref** é um nome de campo da tabela carregada.
- **operator1** é um operador unário (atuando em uma expressão, a da direita).
- **operator2** é um operador binário (atuando em duas expressões, uma de cada lado).
- **function ::= functionname(parameters)**

- **parameters ::= expression { , expression }**

O número e os tipos de parâmetros não são arbitrários. Eles dependem da função utilizada.

Expressões e funções podem, dessa forma, ser aninhadas livremente e, desde que a expressão retorne um valor que possa ser interpretado, o Qlik Sense não apresentará mensagens de erro.

3 Expressões de gráfico

Uma expressão de gráfico (visualização) é uma combinação de funções, campos e operadores matemáticos (+ * / =) e outras medidas. Expressões são utilizadas para processar os dados no aplicativo a fim de produzir um resultado que pode ser visto em uma visualização. Eles não estão limitados à utilização em medidas. É possível criar visualizações mais dinâmicas e eficientes com expressões para títulos, legendas, notas de rodapé e até mesmo dimensões.

Por exemplo, isso significa que em vez do título de uma visualização ser um texto estático, ele pode ser feito a partir de uma expressão cujo resultado é alterado, dependendo das seleções feitas.

Para obter referência detalhada sobre funções de script e funções de gráfico, consulte o Sintaxe de scripts e funções de gráficos.

3.1 Definindo o escopo de agregação

Em geral, existem dois fatores que, juntos, determinam quais registros são usados para definir o valor da agregação em uma expressão. Ao trabalhar com visualizações, esses fatores são:

- Valor dimensional (de agregação em um gráfico de expressão)
- Seleções

Juntos, esses fatores definem o escopo da agregação. Você pode se deparar com situações em que deseja que o cálculo desconsidere a seleção, a dimensão ou ambas. Nas funções de gráfico, você pode conseguir isso usando o qualificador TOTAL, uma análise de conjunto ou uma combinação dos dois.

Agregação: Método e descrição

Método	Descrição
Qualificador TOTAL	Usar o qualificador total dentro de sua função de agregação desconsidera o valor dimensional. A agregação será realizada em todos os valores de campo possíveis. O qualificador **TOTAL** pode vir seguido de uma lista de um ou mais nomes de campos dentro de sinais de maior e menor que. Esses nomes de campos devem ser um subconjunto das variáveis de dimensões do gráfico. Nesse caso, o cálculo é feito ignorando-se todas as variáveis de dimensões do gráfico, exceto aquelas listadas, isto é, um valor será retornado para cada combinação de valores de campo nos campos de dimensão listados. Também podem ser incluídos na lista os campos que não forem uma dimensão em um gráfico no momento. Isso pode ser útil para grupos de dimensões em que os campos de dimensões não são fixos. A lista de todas as variáveis do grupo faz com que a função tenha efeito quando o nível hierárquico for alterado.
Análise de conjunto	Usar a análise de conjunto dentro de sua agregação substitui a seleção. A agregação será realizada em todos os valores divididos através das dimensões.
Qualificador TOTAL e análise de conjunto	Usar o qualificador **TOTAL** e a análise de conjunto dentro da sua agregação substitui a seleção e desconsidera as dimensões.

Qualificador ALL	Usar o qualificador **ALL** dentro da sua agregação desconsidera a seleção e as dimensões. O equivalente pode ser obtido com o comando de análise de conjunto {1} e o qualificador **TOTAL** :
	
	`=sum(All Sales)`
	
	`=sum({1} Total Sales)`

Exemplo: qualificador TOTAL

O exemplo a seguir mostra como TOTAL pode ser utilizado para calcular uma ação relativa. Supondo-se que Q2 tenha sido selecionado, usar TOTAL calcula a soma de todos os valores, desconsiderando as dimensões.

Exemplo: Qualificador total

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%

*Para mostrar os números como uma porcentagem, no painel de propriedades, para a medida que você deseja mostrar como um valor de porcentagem, em **Formatação numérica**, selecione **Número** e, em **Formatação**, escolha **Simples** e um dos formatos de %.*

Exemplo: Análise de conjunto

O exemplo a seguir mostra como a análise de conjunto pode ser usada para fazer uma comparação entre os conjuntos de dados antes de fazer qualquer seleção. Supondo que Q2 tenha sido selecionado, usar a análise de conjunto com a definição de conjunto {1} calcula a soma de todos os valores, desconsiderando as seleções, mas divididos pelas dimensões.

Exemplo: Análise de conjunto

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%

Exemplo: qualificador TOTAL e análise de conjunto

O exemplo a seguir mostra como a análise de conjunto e o qualificador TOTAL podem ser combinados para fazer uma comparação entre os conjuntos de dados antes que qualquer seleção seja feita e entre todas as dimensões. Supondo que Q2 tenha sido selecionado, usar a análise de conjunto com a definição {1} e o qualificador TOTAL calcula a soma de todos os valores, desconsiderando as seleções e as dimensões.

Exemplo: qualificador TOTAL e análise de conjunto

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%

Dados usados nos exemplos:

```
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 ' ');
```

3.2 Análise de conjunto

Ao fazer uma seleção em um aplicativo, você define um subconjunto de registros nos dados. Funções de agregação, como `Sum()`, `Max()`, `Min()`, `Avg()` e `Count()`, são calculadas com base nesse subconjunto.

Em outras palavras, sua seleção define o escopo da agregação, que define o conjunto de registros em que os cálculos são feitos.

A análise de conjunto oferece uma maneira de definir um escopo diferente do conjunto de registros definido pela seleção atual. Esse novo escopo também pode ser considerado uma seleção alternativa.

Isso pode ser útil se você deseja comparar a seleção atual com um valor específico, por exemplo, o valor do ano passado ou a participação no mercado global.

Expressões de conjunto

Expressões de conjunto podem ser usadas dentro e fora das funções de agregação e são colocadas entre colchetes.

Exemplo: Expressão de conjunto interna

```
Sum( {$<Year={2021}>} Sales )
```

Exemplo: Expressão de conjunto externa

```
{<Year={2021}>} Sum(Sales) / Count(distinct Customer)
```

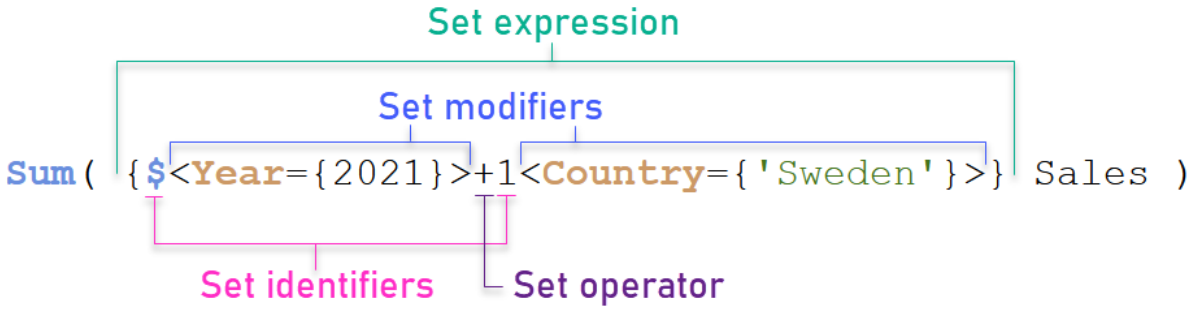
Uma expressão de conjunto consiste em uma combinação dos seguintes elementos:

- **Identificadores**. Um identificador de conjunto representa uma seleção, definida em outro lugar. Também representa um conjunto específico de registros nos dados. Pode ser a seleção atual, uma seleção de um marcador ou uma seleção de um estado alternado. Uma expressão de conjunto consiste de um identificador único, como o sinal de dólar, {$}, que significa todos os registros na seleção atual.
  Exemplos: `$`, `1`, `BookMark1`, `State2`
- **Operadores**. Um operador de conjunto pode ser usado para criar uniões, diferenças ou interseções entre diferentes identificadores de conjunto. Dessa forma, você pode criar um subconjunto ou superconjunto das seleções definidas pelos identificadores de conjunto.
  Exemplos: `+`, `-`, `*`, `/`

- **Modificadores**. Um modificador de conjunto pode ser adicionado ao identificador de conjunto para alterar sua seleção. Um modificador também pode ser usado sozinho e, dessa forma, modificará o identificador padrão. Um modificador deve ser colocado entre sinais de maior e menor <...>. Exemplos: `<Year={2020}>`, `<Supplier={ACME}>`

Os elementos são combinados para formar expressões de conjunto.

Elementos em uma expressão de conjunto



A expressão de conjunto acima, por exemplo, é construída a partir da agregação `Sum(Sales)`.

O primeiro operando retorna as vendas do ano `2021` para a seleção atual, que é indicada pelo identificador de conjunto `$` e pelo modificador que contém a seleção do ano `2021`. O segundo operando retorna `Sales` para `Sweden` e ignora a seleção atual, que é indicada pelo identificador de conjunto `1`.

Por fim, a expressão retorna um conjunto que consiste nos registros que pertencem a qualquer um dos dois operandos de conjunto, conforme indicado pelo operador de conjunto `+`.

Exemplos

Exemplos que combinam os elementos da expressão de conjunto acima estão disponíveis nos seguintes tópicos:

Conjuntos naturais

Normalmente, uma expressão de conjunto representa um conjunto de registros no modelo de dados e uma seleção que define esse subconjunto de dados. Nesse caso, o conjunto se chama conjunto natural.

Identificadores de conjunto, com ou sem modificadores de conjunto, sempre representam conjuntos naturais.

No entanto, uma expressão de conjunto usando operadores de conjunto também representa um subconjunto dos registros, mas, em geral, ainda não pode ser descrita usando uma seleção de valores de campos. Essa expressão é um conjunto não natural.

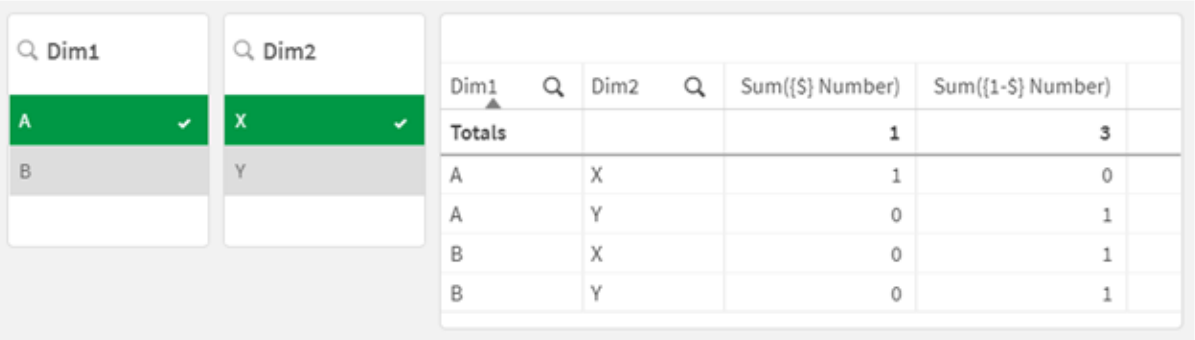
Por exemplo, o conjunto especificado por `{1-$}` nem sempre pode ser definido por uma seleção. Portanto, não é um conjunto natural. Isso pode ser mostrado carregando os dados a seguir, adicionando-os a uma tabela e, em seguida, fazendo seleções com o uso de painéis de filtro.

```
Load * Inline
[Dim1, Dim2, Number
A, X, 1
```

```
A, Y, 1
B, X, 1
B, Y, 1];
```

Fazendo seleções para `Dim1` e `Dim2`, você obtém a exibição mostrada na tabela a seguir.

Tabela com conjuntos naturais e não naturais



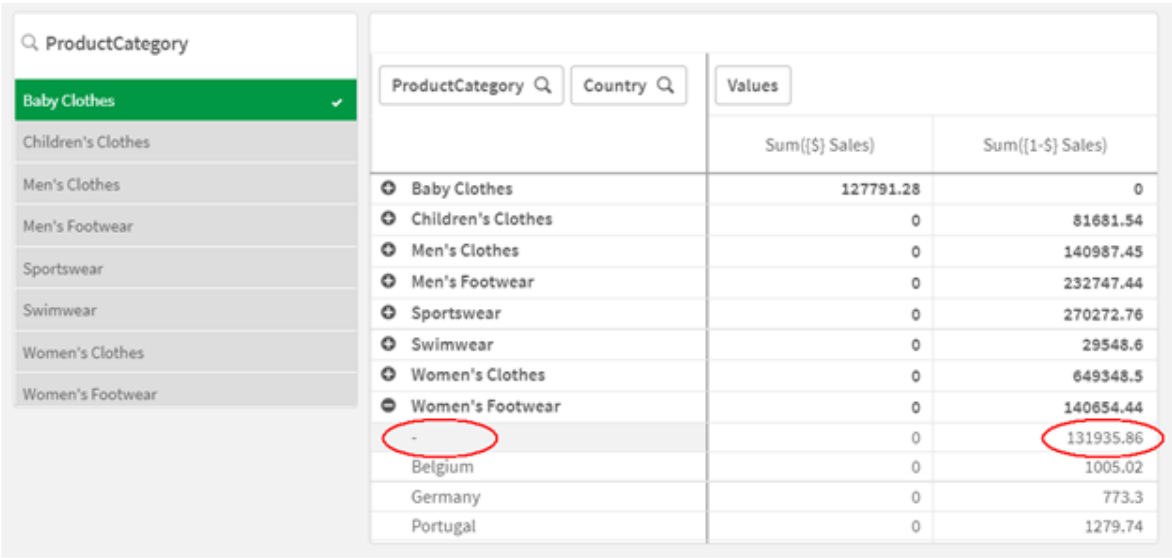
A expressão de conjunto na primeira medida usa um conjunto natural. Ela corresponde à seleção que é feita `{$}`.

A segunda medida é diferente. Ela usa `{1-$}`. Não é possível fazer uma seleção que corresponda a esse conjunto e, portanto, ele é um conjunto não natural.

Essa distinção tem uma série de consequências:

- Modificadores de conjunto só podem ser aplicados a identificadores de conjunto. Eles não podem ser aplicados a uma expressão de conjunto arbitrária. Por exemplo, não é possível usar uma expressão de conjunto como:
  `{ (BM01 * BM02) <Field={x,y}> }`
  Aqui, os colchetes normais (redondos) implicam que a interseção entre `BM01` e `BM02` deve ser avaliada antes que o modificador de conjunto seja aplicado. A razão é que não há um conjunto de elementos que possa ser modificado.
- Você não pode usar conjuntos não naturais dentro de funções de elementos `P()` e `E()`. Essas funções retornam um conjunto de elementos, mas não é possível deduzir o conjunto de elementos de um conjunto não natural.
- Uma medida que usa um conjunto não natural nem sempre pode ser atribuída ao valor dimensional correto quando o modelo de dados tem muitas tabelas. Por exemplo, no gráfico a seguir, alguns números de vendas excluídos são atribuídos ao Country correto, enquanto outros têm NULL como Country.

Gráfico com conjunto não natural



O que determina se a atribuição é feita ou não corretamente depende do modelo de dados. Nesse caso, o número não poderá ser atribuído se pertencer a um país que foi excluído pela seleção.

Identificador	Descrição
1	Representa um conjunto completo de todos os registros no aplicativo, independente de quaisquer seleções feitas.
$	Representa os registros da seleção atual. A expressão de conjunto **{$}** é equivalente a não definir uma expressão de conjunto.
$1	Representa a seleção anterior. $2 representa seleção-menos-um anterior, e adiante.
$_1	Representa a próxima (para frente) seleção. $_2 representa a próxima seleção-mais-um, e adiante.
BM01	Você pode usar qualquer ID de marcador ou nome de marcador.
MyAltState	Você pode fazer referência às seleções realizadas em um estado alternado por seu nome de estado.

Exemplo	Resultado
sum ({1} Sales)	Retorna o total das vendas no aplicativo, desconsiderando as seleções porém não a dimensão.
sum ({$} Sales)	Retorna as vendas da seleção atual, ou seja, o mesmo que sum(Sales).
sum ({$1} Sales)	Retorna as vendas da seleção anterior.
sum ({BM01} Sales)	Retorna as vendas do marcador chamado *BM01*.

Exemplo	Resultado
sum({$<OrderDate = DeliveryDate>} Sales)	Retorna as vendas da seleção atual onde OrderDate = DeliveryDate.
sum({1<Region = {US}>} Sales)	Retorna as vendas da região dos EUA, desconsiderando a seleção atual.
sum({$<Region = >} Sales)	Retorna as vendas da seleção, mas com a seleção em '*Region*' removida.
sum({<Region = >} Sales)	Retorna o mesmo como o exemplo acima. Quando o conjunto para modificar for omitido, $ é adotado.
sum({$<Year={2000}, Region={“U*”}>} Sales)	Retorna as vendas da seleção atual, mas com novas seleções ambas em '*Year*' e em '*Region*'.

Identificadores de conjunto

Um identificador de conjunto representa um conjunto de registros nos dados: todos os dados ou um subconjunto dos dados. É o conjunto de registros definido por uma seleção. Pode ser a seleção atual, todos os dados (sem seleção), uma seleção de um marcador ou uma seleção de um estado alternado.

No exemplo `Sum( {$<Year = {2009}>} Sales )`, o identificador é o sinal de cifrão: `$`. Isso representa a seleção atual. Também representa todos os registros possíveis. Esse conjunto pode então ser alterado pela parte modificadora da expressão de conjunto: a seleção `2009` em `Year` é adicionada.

Em uma expressão de conjunto mais complexa, dois identificadores podem ser usados juntos com um operador para formar uma união, uma diferença ou uma interseção dos dois conjuntos de registros.

A tabela a seguir mostra alguns identificadores comuns.

Exemplos com identificadores comuns

Identificador	Descrição
`1`	Representa um conjunto completo de todos os registros no aplicativo, independente de quaisquer seleções feitas.
`$`	Representa os registros da seleção atual no estado padrão. Portanto, a expressão de conjunto `{$}` equivale de modo geral a não declarar uma expressão de conjunto.
`$1`	Representa a seleção anterior no estado padrão. `$2` representa a seleção anterior, menos um, e assim por diante.
`$_1`	Representa a próxima seleção (para frente). `$_2` representa a próxima seleção, menos um, e assim por diante.
`BM01`	Você pode usar qualquer ID de marcador ou nome de marcador.
`AltState`	Você pode fazer referência a um estado alternado por seu nome de estado.
`AltState::BM01`	Um marcador contém as seleções de todos os estados, e você pode fazer referência a um marcador específico qualificando o nome desse marcador.

A tabela a seguir mostra exemplos com diferentes identificadores.

Exemplos com identificadores diferentes

Exemplo	Resultado
`Sum ({1} Sales)`	Retorna o total das vendas no aplicativo, desconsiderando as seleções, mas não a dimensão.
`Sum ({$} Sales)`	Retorna as vendas da seleção atual , ou seja , o mesmo que `Sum(Sales)`.
`Sum ({$1} Sales)`	Retorna as vendas da seleção anterior.
`Sum ({BM01} Sales)`	Retorna as vendas do marcador chamado `BM01`.

Operadores de conjunto

Operadores de conjunto são usados para incluir, excluir ou cruzar conjuntos de dados. Todos os operadores usam conjuntos como operandos e retornam um conjunto como resultado.

Você pode usar operadores de conjunto em duas situações diferentes:

- Para executar uma operação de conjunto em identificadores de conjunto, representando conjuntos de registros em dados.
- Para executar uma operação de conjunto nos conjuntos de elementos, nos valores de campo ou dentro de um modificador de conjunto.

A tabela a seguir mostra os operadores que podem ser usados em expressões de conjunto.

Operadores

Operador	Descrição
`+`	União. Essa operação binária retorna um conjunto que consiste nos registros ou elementos que pertencem a qualquer um dos dois operandos de conjunto.
`-`	Exclusão. Essa operação binária retorna um conjunto que consiste nos registros ou elementos que pertencem ao primeiro, mas não ao outro dos dois operandos de conjunto. Além disso, quando usada como um operador unário, ela retorna o conjunto complementar.
`*`	Interseção. Essa operação binária retorna um conjunto que consiste nos registros ou elementos que pertencem a ambos os operandos de conjunto.
`/`	Diferença simétrica (`XOR`). Essa operação binária retorna um conjunto que consiste nos registros ou elementos que pertencem a um dos operandos de conjunto, mas não a ambos.

A tabela a seguir mostra exemplos com operadores.

Exemplos com operadores

Exemplo	Resultado
`Sum ({1-$} Sales)`	Retorna as vendas para tudo excluído pela seleção atual.
`Sum ({$*BM01} Sales)`	Retorna as vendas da interseção entre a seleção e o marcador #160;`BM01`.

Exemplo	Resultado
`Sum ({-($+BM01)} Sales)`	Retorna as vendas excluídas pela seleção e o marcador `BM01`.
`Sum ({$<Year= {2009}>+1<Country= {'Sweden'}>} Sales)`	Retorna as vendas do ano de 2009 associadas às `seleções` atuais e adiciona o conjunto completo de dados associados ao país Sweden em todos os anos.
`Sum ({$<Country={"S*"}+ {"*land"}>} Sales)`	Retorna as vendas para países que começam com `S` ou terminam com `land`.

Modificadores de conjunto

Expressões de conjunto são usadas para definir o escopo de um cálculo. A parte central da expressão de conjunto é o modificador de conjunto que especifica uma seleção. Isso é usado para modificar a seleção do usuário, ou a seleção no identificador de conjunto, e o resultado define um novo escopo para o cálculo.

O modificador de conjunto consiste em um ou mais nomes de campo, cada um seguido por uma seleção que deve ser feita no campo. O modificador está entre sinais de maior e menor: `< >`

Por exemplo:

- `Sum ( {$<Year = {2015}>} Sales )`
- `Count ( {1<Country = {Germany}>} distinct OrderID )`
- `Sum ( {$<Year = {2015}, Country = {Germany}>} Sales )`

Conjuntos de elementos

Um conjunto de elementos pode ser definido usando o seguinte:

- Uma lista de valores
- Uma pesquisa
- Uma referência a outro campo
- Uma função de conjunto

Se a definição do conjunto de elementos for omitida, o modificador de conjunto apagará qualquer seleção nesse campo. Por exemplo:

```
Sum( {$<Year = >} Sales )
```

Exemplos: Expressões de gráfico para modificadores de conjunto com base em conjuntos de elementos

Exemplos - expressões de gráfico

Script de carregamento

Carregue os seguintes dados como um carregamento inline no editor de carregamento de dados para criar os exemplos de expressão de gráfico abaixo.

```
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
];
```

Expressões de gráfico

Crie uma tabela em uma pasta do Qlik Sense com as seguintes expressões de gráfico.

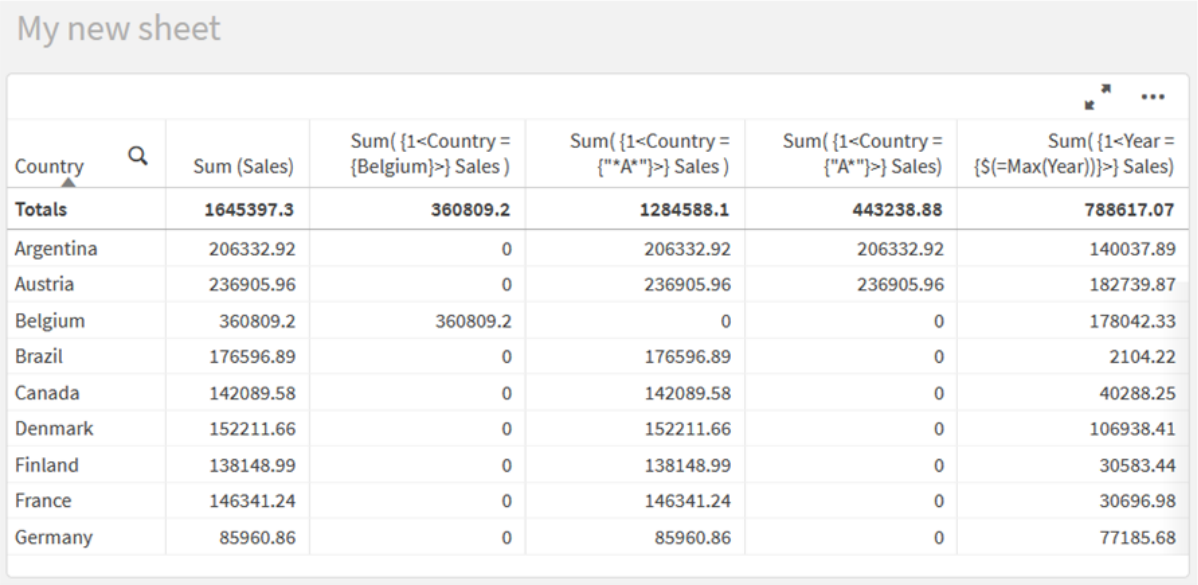
Tabela - Modificadores de conjunto com base em conjuntos de elementos

Country	**Sum(Sales)**	**Sum ({1<Country= {Belgium}>} Sales)**	**Sum ({1<Country= {"*A*"}>} Sales)**	**Sum ({1<Country= {"A*"}>} Sales)**	**Sum({1<Year= {$(=Max (Year))}>} Sales)**
Totais	1645397.3	360809.2	1284588.1	443238.88	788617.07
Argentina	206332.92	0	206332.92	206332.92	140037.89
Áustria	236905.96	0	236905.96	236905.96	182739.87
Bélgica	360809.2	360809.2	0	0	178042.33
Brasil	176596.89	0	176596.89	0	2104.22
Canadá	142089.58	0	142089.58	0	40288.25
Dinamarca	152211.66	0	152211.66	0	106938.41
Finlândia	138148.99	0	138148.99	0	30583.44
França	146341.24	0	146341.24	0	30696.98
Alemanha	85960.86	0	85960.86	0	77185.68

Explicação

- Dimensões:
  - `Country`
- Medidas:
  - `Sum(Sales)`
    Soma `Sales`, sem expressão de conjunto.
  - `Sum({1<Country={Belgium}>}Sales)`
    Selecione `Belgium` e depois some as `Sales` correspondentes.
  - `Sum({1<Country={"*A*"}>}Sales)`
    Selecione todos os países que têm `A` e depois some as `Sales` correspondentes.
  - `Sum({1<Country={"A*"}>}Sales)`
    Selecione todos os países que começam com `A` e depois some as `Sales` correspondentes.
  - `Sum({1<Year={$(=Max(Year))}>}Sales)`
    Calcule `Max(Year)`, que é `2015`, e depois some as `Sales` correspondentes.

Modificadores de conjunto com base em conjuntos de elementos


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

Valores listados

O exemplo mais comum de um conjunto de elementos é aquele baseado em uma lista de valores de campo entre chaves. Por exemplo:

- `{$<Country = {Canada, Germany, Singapore}>}`
- `{$<Year = {2015, 2016}>}`

As chaves internas definem o conjunto de elementos. Os valores individuais são separados por vírgulas.

Aspas e distinção entre maiúsculas e minúsculas

Se os valores contiverem espaços em branco ou caracteres especiais, eles deverão ser colocados entre aspas. As aspas simples serão uma correspondência literal, com distinção entre maiúsculas e minúsculas e um único valor de campo. As aspas duplas significam uma correspondência sem distinção entre maiúsculas e minúsculas e com um ou vários valores de campo. Por exemplo:

- `<Country = {'New Zealand'}>`
  Corresponde `New Zealand` apenas.
- `<Country = {"New Zealand"}>`
  Corresponde `New Zealand`, `NEW ZEALAND` e `new zealand`.

As datas devem ser colocadas entre aspas e usar o formato de data do campo em questão. Por exemplo:

- `<ISO_Date = {'2021-12-31'}>`
- `<US_Date = {'12/31/2021'}>`
- `<UK_Date = {'31/12/2021'}>`

Aspas duplas podem ser substituídas por colchetes ou acentos graves.

Pesquisas

Cconjuntos de elementos também podem ser criados por meio de pesquisas. Por exemplo:

- `<Country = {"C*"}>`
- `<Ingredient = {"*garlic*"}>`
- `<Year = {">2015"}>`
- `<Date = {">12/31/2015"}>`

Curingas podem ser usados em pesquisas de texto: Um asterisco (*) representa qualquer número de caracteres, e um ponto de interrogação (?) representa um único caractere. Operadores relacionais podem ser usados para definir pesquisas numéricas.

Você sempre deve usar aspas duplas para pesquisas. As pesquisas diferenciam maiúsculas e minúsculas.

Expansões de dólar

Expansões de dólares serão necessárias se você quiser usar um cálculo dentro do seu conjunto de elementos. Por exemplo, se quiser ver apenas o último ano possível, use:

```
<Year = {$(=Max(Year))}>
```

Valores selecionados em outros campos

Modificadores podem ser baseados nos valores selecionados de outro campo. Por exemplo:

```
<OrderDate = DeliveryDate>
```

Esse modificador usará os valores selecionados de `DeliveryDate` e os aplicará como uma seleção em `OrderDate`. Se houver muitos valores diferentes – mais de duzentos – esta operação ocupará muito a CPU e deverá ser evitada.

Funções do conjunto de elementos

O conjunto de elementos também pode ser baseado nas funções de conjuntos `P()` (valores possíveis) e `E()` (valores excluídos).

Por exemplo, se quiser selecionar países em que o produto `Cap` foi vendido, use:

```
<Country = P({1<Product={Cap}>} Country)>
```

Da mesma forma, se quiser escolher os países onde o produto `Cap` não foi vendido, use:

```
<Country = E({1<Product={Cap}>} Country)>
```

Modificadores de conjunto com pesquisas

Você pode criar conjuntos de elementos por meio de pesquisas com modificadores de conjunto.

Por exemplo:

- `<Country = {"C*"}>`
- `<Year = {">2015"}>`
- `<Ingredient = {"*garlic*"}>`

Pesquisas devem sempre ser colocadas entre aspas duplas, colchetes ou acentos graves. Você pode usar uma lista com uma combinação de strings literais (aspas simples) e pesquisas (aspas duplas). Por exemplo:

```
<Product = {'Nut', "*Bolt", Washer}>
```

Pesquisas de texto

Curingas e outros símbolos podem ser usados em pesquisas de texto:

- Um asterisco (*) representará qualquer número de caracteres.
- Um ponto de interrogação (?) representará um único caractere.
- Um acento circunflexo (^) marcará o início de uma palavra.

Por exemplo:

- `<Country = {"C*", "*land"}>`
  Corresponda todos os países que começam com `C` ou terminam com `land`.
- `<Country = {"*^z*"}>`
  Isso corresponderá todos os países que têm uma palavra que começa com `z`, como `New Zealand`.

Pesquisas numéricas

Você pode fazer pesquisas numéricas usando estes operadores relacionais: >, >=, <, <=

Uma pesquisa numérica sempre começa com um desses operadores. Por exemplo:

- `<Year = {">2015"}>`
  Corresponda 2016 e anos subsequentes.
- `<Date = {">=1/1/2015<1/1/2016"}>`

Corresponda todas as datas durante 2015. Observe a sintaxe para descrever um intervalo de tempo entre duas datas. O formato da data deve corresponder ao formato da data do campo em questão.

Pesquisas de expressão

Você pode usar pesquisas de expressão para fazer pesquisas mais avançadas. Uma agregação é então avaliada para cada valor de campo no campo de pesquisa. Todos os valores para os quais a expressão de pesquisa retorna "true" são selecionados.

Uma pesquisa de expressão sempre começa com um sinal de igual: =

Por exemplo:

```
<Customer = {"=Sum(Sales)>1000"}>
```

Isso retornará todos os clientes com um valor de vendas superior a 1000. `Sum(Sales)` é calculado na seleção atual. Isso significa que, se você tiver uma seleção em outro campo, como o campo `Product`, obterá os clientes que atenderam à condição de vendas apenas para os produtos selecionados.

Se quiser que a condição seja independente da seleção, você precisará usar a análise de conjunto nos caracteres de pesquisa. Por exemplo:

```
<Customer = {"=Sum({1} Sales)>1000"}>
```

As expressões após o sinal de igual serão interpretadas como um valor booleano. Isso significa que, se ela for avaliada como algo diferente, qualquer número diferente de zero será interpretado como "true", enquanto zero e sequências de caracteres serão interpretados como "false".

Quotes

Use aspas quando os caracteres de busca contiverem espaços em branco ou sequências de caracteres especiais. As aspas simples implicam uma correspondência literal, com distinção entre maiúsculas e minúsculas e com um único valor de campo. As aspas duplas implicam uma pesquisa sem distinção entre maiúsculas e minúsculas que corresponde potencialmente a vários valores de campos.

Por exemplo:

- `<Country = {'New Zealand'}>`
  Corresponde `New Zealand` apenas.
- `<Country = {"New Zealand"}>`
  Corresponde `New Zealand`, `NEW ZEALAND` e `new zealand`

Aspas duplas podem ser substituídas por colchetes ou acentos graves.

Nas versões anteriores do Qlik Sense, não havia distinção entre aspas simples e aspas duplas, e todas as sequências entre aspas eram tratadas como pesquisas. Para manter a compatibilidade com versões anteriores, os aplicativos criados com versões mais antigas do Qlik Sense continuarão a funcionar como faziam em versões anteriores. Aplicativos criados com o Qlik Sense November 2017 ou versões posteriores respeitarão a diferença entre os dois tipos de aspas.

Exemplos: Expressões de gráfico para modificadores de conjunto com pesquisas

Exemplos - expressões de gráfico

Script de carregamento

Carregue os seguintes dados como um carregamento inline no editor de carregamento de dados para criar os exemplos de expressão de gráfico abaixo.

```
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];
```

Exemplo 1: Expressões de gráfico com pesquisas de texto

Crie uma tabela em uma pasta do Qlik Sense com as seguintes expressões de gráfico.

Tabela - Modificadores de conjunto com pesquisas de texto

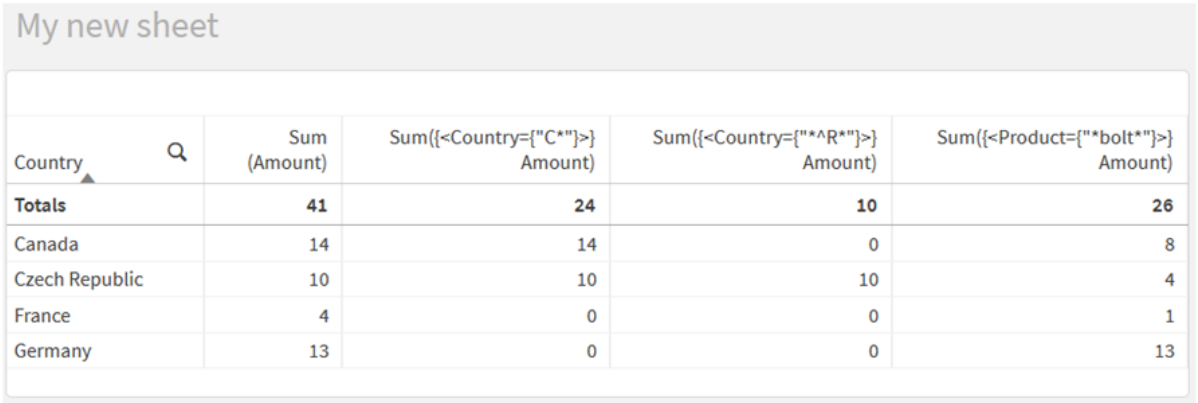
Country	Sum (Amount)	Sum({<Country={"C*"}>} Amount)	Sum({<Country={"*^R*"}>} Amount)	Sum({<Product={"*bolt*"}>} Amount)
Totais	**41**	**24**	**10**	**26**
Canadá	14	14	0	8
República Tcheca	10	10	10	4
França	4	0	0	1
Alemanha	13	0	0	13

Explicação

- Dimensões:
    - `Country`

- Medidas:
  - `Sum(Amount)`
    Soma `Amount`, sem expressão de conjunto.
  - `Sum({<Country={"C*"}>}Amount)`
    Soma `Amount` para todos os países que começam com `C`, como `Canada` e `Czech Republic`.
  - `Sum({<Country={"*^R*"}>}Amount)`
    Soma `Amount` para todos os países que têm uma palavra que começa com `R`, como `Czech Republic`.
  - `Sum({<Product={"*bolt*"}>}Amount)`
    Soma `Amount` para todos os produtos que contêm a sequência de caracteres `bolt`, como `Bolt` e `Anchor bolt`.

Modificadores de conjunto com pesquisas de texto



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

Exemplo 2: Expressões de gráfico com pesquisas numéricas

Crie uma tabela em uma pasta do Qlik Sense com as seguintes expressões de gráfico.

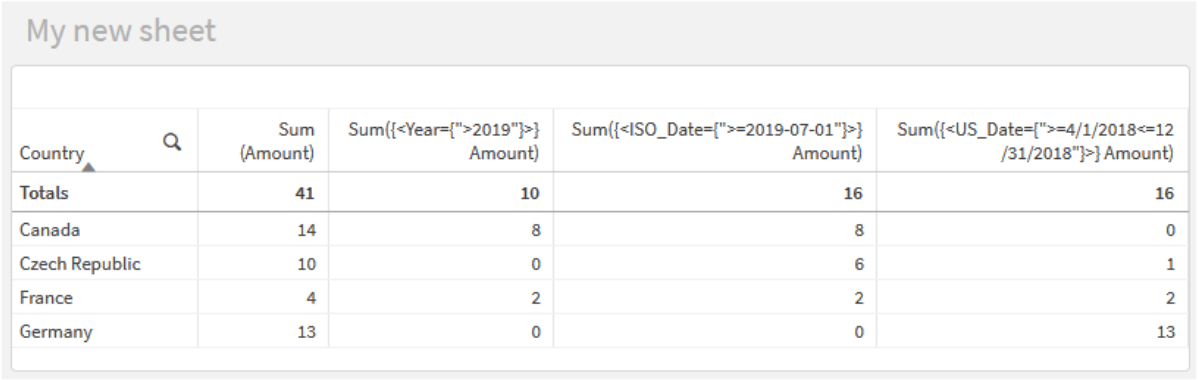
Tabela - Modificadores de conjunto com pesquisas numéricas

Country	Sum (Amount)	Sum({<Year={">2019"}>} Amount)	Sum({<ISO_Date={">=2019-07-01"}>} Amount)	Sum({<US_Date={">=4/1/2018<=12/31/2018"}>} Amount)
Totais	**41**	**10**	**16**	**16**
Canadá	14	8	8	0
República Tcheca	10	0	6	1
França	4	2	2	2
Alemanha	13	0	0	13

Explicação

- Dimensões:
  - `Country`
- Medidas:
  - `Sum(Amount)`
    Soma `Amount`, sem expressão de conjunto.
  - `Sum({<Year={">2019"}>}Amount)`
    Soma `Amount` por todos os anos após 2019.
  - `Sum({<ISO_Date={">=2019-07-01"}>}Amount)`
    Soma `Amount` para todas as datas em ou depois de `2019-07-01`. O formato da data na pesquisa deve corresponder ao formato do campo.
  - `Sum({<US_Date={">=4/1/2018<=12/31/2018"}>}Amount)`
    Soma `Amount` para todas as datas de `4/1/2018` a `12/31/2018`, incluindo as datas de início e de término. O formato das datas na pesquisa deve corresponder ao formato do campo.

Modificadores de conjunto com pesquisas numéricas



Exemplo 3: Expressões de gráfico com pesquisas de expressão

Crie uma tabela em uma pasta do Qlik Sense com as seguintes expressões de gráfico.

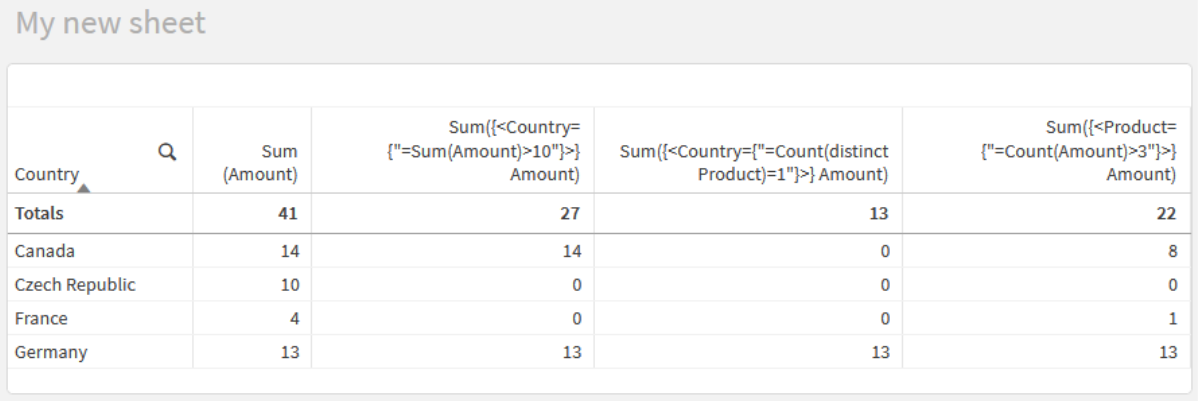
Table - Set modifiers with expression searches

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

Explicação

- Dimensões:
  - `Country`
- Medidas:
  - `Sum(Amount)`
    Soma `Amount`, sem expressão de conjunto.
  - `Sum({<Country={"=Sum(Amount)>10"}>}Amount)`
    Soma `Amount` para todos os países que têm uma soma agregada de `Amount` maior que 10.
  - `Sum({<Country={"=Count(distinct Product)=1"}>}Amount)`
    Soma `Amount` para todos os países associados a exatamente um produto distinto.
  - `Sum({<Product={"=Count(Amount)>3"}>}Amount)`
    Soma `Amount` para todos os países que têm mais de três transações nos dados.

Modificadores de conjunto com pesquisas de expressão



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

Exemplos	Resultados
sum({$–1<Product = {“*Internal*”, “*Domestic*”}>} Sales)	Retorna as vendas da seleção atual, excluindo as transações que pertencem a produtos com a cadeia 'Internal' ou 'Domestic' no nome do produto.
sum({$<Customer = {“=Sum ({1<Year = {2007}>} Sales) > 1000000”}>} Sales)	Retorna as vendas da seleção atual, mas com uma nova seleção no campo 'Customer': apenas os clientes que, em 2007, apresentaram um total de vendas de mais de 1.000.000.

Modificadores de conjunto com expansões de sinal de dólar

Expansões de sinal de cifrão são construções calculadas antes que a expressão seja analisada e avaliada. O resultado é então injetado na expressão em vez de `$(…)`. O cálculo da expressão é então feito usando o resultado da expansão de sinal de cifrão.

O editor de expressões mostra uma visualização da expansão de sinal de cifrão para que você possa verificar o que sua expansão de sinal de cifrão avalia.

Visualização de expansão de sinal de dólar no editor de expressões

Use expansões de sinal de dólar quando quiser usar um cálculo dentro do seu conjunto de elementos.

Por exemplo, se quiser ver apenas o último ano possível, use a seguinte construção:

```
<Year = {$(=Max(Year))}>
```

`Max(Year)` é calculado primeiro, e o resultado é injetado na expressão em vez de `$(…)`.

O resultado após a expansão do cifrão será uma expressão como a que se segue:

```
<Year = {2021}>
```

A expressão dentro da expansão de sinal de dólar é calculada com base na seleção atual. Isso significa que, se você tiver uma seleção em outro campo, o resultado da expressão será afetado.

Se quiser que o cálculo seja independente da seleção, use a análise de conjunto dentro da expansão de sinal de dólar. Por exemplo:

```
<Year = {$(=Max({1} Year))}>
```

Sequências de caracteres

Quando você deseja que a expansão de sinal de dólar resulte em uma cadeia de caracteres, as regras normais de aplicação de aspas são aplicáveis. Por exemplo:

```
<Country = {'$(=FirstSortedValue(Country,Date)'}>
```

O resultado após a expansão do cifrão será uma expressão como a que se segue:

```
<Country = {'New Zealand'}>
```

Você obterá um erro de sintaxe se não usar as aspas.

Números

Quando você deseja que a expansão de sinal de dólar resulte em um número, certifique-se de que ela tenha a mesma formatação que o campo. Isso significa que, às vezes, você precisa envolver a expressão em uma função de formatação.

Por exemplo:

```
<Amount = {$(=Num(Max(Amount), '###0.00'))}>
```

O resultado após a expansão do cifrão será uma expressão como a que se segue:

```
<Amount = {12362.00}>
```

Use um hash para forçar a expansão a sempre usar o ponto decimal e nenhum separador de milhar. Por exemplo:

```
<Amount = {$(#=Max(Amount))}>
```

Datas

Quando quiser que a expansão de sinal de dólar resulte em uma data, certifique-se de que ela tenha a formatação correta. Isso significa que, às vezes, você precisa envolver a expressão em uma função de formatação.

Por exemplo:

```
<Date = {'$(=Date(Max(Date)))'}>
```

O resultado após a expansão do cifrão será uma expressão como a que se segue:

```
<Date = {'12/31/2015'}>
```

Assim como com sequências de caracteres, você precisa usar as aspas corretas.

Um caso de uso comum é quando você deseja que seu cálculo seja limitado ao último mês (ou ano). Nesse caso, é possível usar uma pesquisa numérica em combinação com a função `AddMonths()`.

Por exemplo:

```
<Date = {">=$(=AddMonths(Today(),-1))"}>
```

O resultado após a expansão do cifrão será uma expressão como a que se segue:

```
<Date = {">=9/31/2021"}>
```

Isso selecionará todos os eventos que ocorreram no mês passado.

Exemplo: Expressões de gráfico para modificadores de conjunto com expansões de sinal de dólar

Exemplo: expressões de gráfico

Script de carregamento

Carregue os seguintes dados como um carregamento inline no editor de carregamento de dados para criar os exemplos de expressão de gráfico abaixo.

```
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];
```

Expressões de gráfico com expansões de sinal de dólar

Crie uma tabela em uma pasta do Qlik Sense com as seguintes expressões de gráfico.

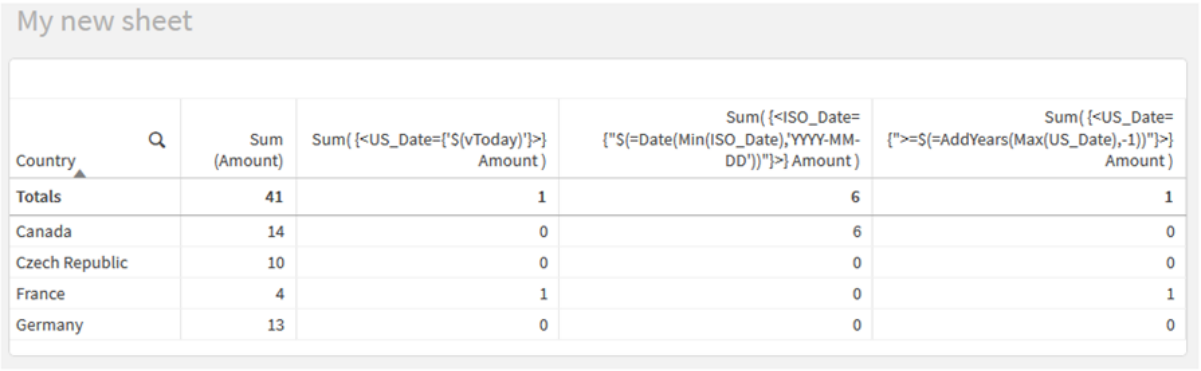
Tabela - Modificadores de conjunto com expansões de sinal de cifrão

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)
Totais	**41**	**1**	**6**	**1**
Canadá	14	0	6	0
República Tcheca	10	0	0	0
França	4	1	0	1
Alemanha	13	0	0	0

Explicação

- Dimensões:
  - `Country`
- Medidas:
  - `Sum(Amount)`
    Soma `Amount` sem uma expressão de conjunto.
  - `Sum({<US_Date={'$(vToday)'}>}Amount)`
    Soma `Amount` para todos os registros em que `US_Date` é igual ao da variável`vToday`.
  - `Sum({<ISO_Date={"$(=Date(Min(ISO_Date),'YYYY-MM-DD'))"}>}Amount)`
    Soma `Amount` para todos os registros em que `ISO_Date` é igual à primeira `ISO_Date` (menor) possível. A função `Date()` é necessária para garantir que o formato da data corresponda ao do campo.
  - `Sum({<US_Date={">=$(=AddYears(Max(US_Date),-1))"}>}Amount)`
    Soma `Amount` para todos os registros que têm `US_Date` após ou na data um ano antes da última `US_Date` (maior) possível. A função `AddYears()` retornará uma data no formato especificado pela variável `DateFormat`, e isso precisa corresponder ao formato do campo `US_Date`.

Modificadores de conjunto com expansões de sinal de cifrão


My new sheet

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

Exemplos	Resultados
sum({$<Year = {$(#vLastYear)}>} Sales)	Retorna as vendas do ano anterior em relação à seleção atual. Aqui, uma variável vLastYear contendo o ano relevante é usada em uma expansão de sinal de dólar.
sum({$<Year = {$(#=Only(Year)-1)}>} Sales)	Retorna as vendas do ano anterior em relação à seleção atual. Aqui, uma expansão de sinal de dólar é usada para calcular o ano anterior.

Modificadores de conjunto com operadores de conjunto

Operadores de conjunto são usados para incluir, excluir ou cruzar diferentes conjuntos de elementos. Eles combinam os diferentes métodos para definir conjuntos de elementos.

Os operadores são iguais aos usados para identificadores de conjunto.

Operadores

Operador	Descrição
+	União. Essa operação binária retorna um conjunto que consiste nos registros ou elementos que pertencem a qualquer um dos dois operandos de conjunto.
-	Exclusão. Essa operação binária retorna um conjunto que consiste nos registros ou elementos que pertencem ao primeiro, mas não ao outro dos dois operandos de conjunto. Além disso, quando usada como um operador unário, ela retorna o conjunto complementar.
*	Interseção. Essa operação binária retorna um conjunto que consiste nos registros ou elementos que pertencem a ambos os operandos de conjunto.
/	Diferença simétrica (XOR). Essa operação binária retorna um conjunto que consiste nos registros ou elementos que pertencem a um dos operandos de conjunto, mas não a ambos.

Por exemplo, os dois modificadores a seguir definem o mesmo conjunto de valores de campo:

- `<Year = {1997, "20*"}>`
- `<Year = {1997} + {"20*"}>`

Ambas as expressões selecionam 1997 e os anos que começam com 20. Em outras palavras, essa é a união das duas condições.

Operadores de conjunto também permitem definições mais complexas. Por exemplo:

```
<Year = {1997, "20*"} - {2000}>
```

Essa expressão selecionará os mesmos anos que os anteriores, mas, além disso, excluirá o ano 2000.

.

Exemplos: Expressões de gráfico para modificadores de conjunto com operadores de conjunto

Exemplos - expressões de gráfico

Script de carregamento

Carregue os seguintes dados como um carregamento inline no editor de carregamento de dados para criar os exemplos de expressão de gráfico abaixo.

```
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];
```

Expressões de gráfico

Crie uma tabela em uma pasta do Qlik Sense com as seguintes expressões de gráfico.

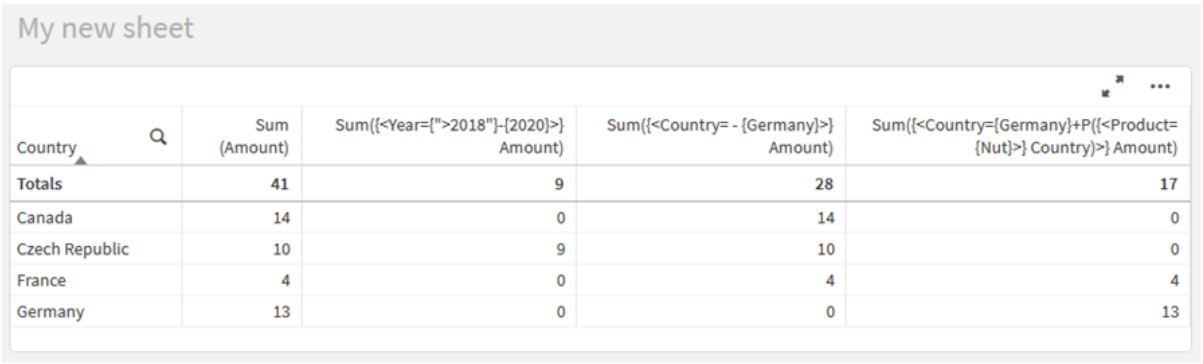
Tabela - Modificadores de conjunto com operadores de conjunto

Country	Sum (Amount)	Sum({<Year={">2018"}-{2020}>} Amount)	Sum({<Country=-{Germany}>} Amount)	Sum({<Country={Germany}+P({<Product={Nut}>}Country)>} Amount)
Totais	**41**	**9**	**28**	**17**
Canadá	14	0	14	0
República Tcheca	10	9	10	0
França	4	0	4	4
Alemanha	13	0	0	13

Explicação

- Dimensões:
  - `Country`
- Medidas:
  - `Sum(Amount)`
    Soma `Amount`, sem expressão de conjunto.
  - `Sum({<Year={">2018"}-{2020}>}Amount)`
    Soma `Amount` para todos os anos depois de 2018, exceto 2020.
  - `Sum({<Country=-{Germany}>}Amount)`
    Soma `Amount` para todos os países, exceto `Germany`. Observe o operador de exclusão unário.
  - `Sum({<Country={Germany}+P({<Product={Nut}>}Country)>}Amount)`
    Soma `Amount` para `Germany` e todos os países associados ao produto `Nut`.

Modificadores de conjunto com operadores de conjunto



Exemplos	Resultados
sum({$<Product = Product + {OurProduct1} – {OurProduct2} >} Sales)	Retorna as vendas da seleção atual, mas com o produto “OurProduct1” adicionado à lista de produtos selecionados e “OurProduct2” removido da lista de produtos selecionados.
sum({$<Year = Year + ({“20*”,1997} – {2000}) >} Sales)	Retorna as vendas da seleção atual, mas com seleções adicionais no campo “Year”: 1997 e todos os anos que começam com “20”, exceto o ano 2000. Observe que se 2000 estivesse incluído na seleção atual, ele continuaria incluído após a modificação.
sum({$<Year = (Year + {“20*”,1997}) – {2000} >} Sales)	Retorna quase o mesmo resultado acima, mas aqui o ano 2000 será excluído, mesmo se estivesse inicialmente incluído na seleção atual. O exemplo mostra a importância da utilização de parênteses em alguns casos, para definir uma ordem de precedência.
sum({$<Year = {“*”} – {2000}, Product = {“*bearing*”} >} Sales)	Retorna as vendas da seleção atual, mas com uma nova seleção em “Year”: todos os anos exceto o ano 2000; e somente para produtos que contêm a palavra 'suporte'.

Modificadores de conjunto com operadores de conjunto implícitos

A maneira padrão de escrever seleções em um modificador de conjunto é usar um sinal de igual. Por exemplo:

```
Year = {">2015"}
```

A expressão à direita do sinal de igual no modificador de conjunto é chamada de conjunto de elementos. Ela define um conjunto de valores de campo distintos, em outras palavras, uma seleção.

Essa notação define uma nova seleção, desconsiderando a seleção atual no campo. Portanto, se o identificador do conjunto contiver uma seleção nesse campo, a seleção antiga será substituída por aquela do conjunto de elementos.

Quando você quiser basear sua seleção na seleção atual no campo, precisará usar uma expressão diferente

Por exemplo, se quiser respeitar a seleção antiga e adicionar o requisito de que o ano seja posterior a 2015, escreva o seguinte:

```
Year = Year * {">2015"}
```

O asterisco é um operador de conjunto que define uma interseção, então você obterá a interseção entre a seleção atual em `Year` e o requisito adicional de que o ano seja posterior a `2015`. Uma maneira alternativa de escrever isso é:

```
Year *= {">2015"}
```

Ou seja, o operador de atribuição (*=) define implicitamente uma interseção.

Da mesma forma, uniões implícitas, exclusões e diferenças simétricas podem ser definidas usando o seguinte: +=, -=, /=

Exemplos: Expressões de gráfico para modificadores de conjunto com operadores de conjunto implícitos

Exemplos - expressões de gráfico

Script de carregamento

Carregue os seguintes dados como um carregamento inline no editor de carregamento de dados para criar os exemplos de expressão de gráfico abaixo.

```
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];
```

Expressões de gráfico com operadores de conjunto implícitos

Crie uma tabela em uma pasta do Qlik Sense com as seguintes expressões de gráfico.

Selecione `Canada` e `Czech Republic` em uma lista de países.

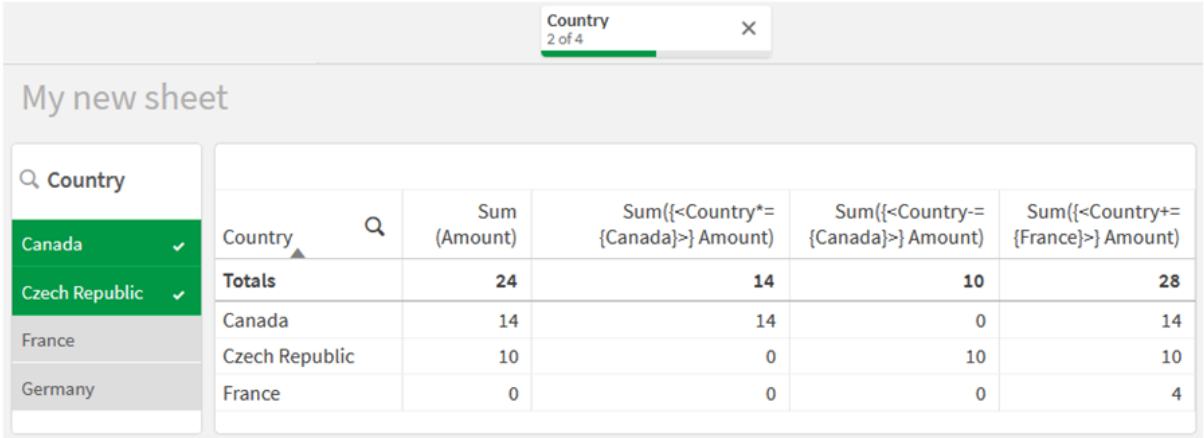
Tabela - Expressões de gráfico com operadores de conjunto implícitos

Country	Sum (Amount)	Sum({<Country*= {Canada}>} Amount)	Sum({<Country-= {Canada}>} Amount)	Sum({<Country+= {France}>} Amount)
Totais	**24**	**14**	**10**	**28**
Canadá	14	14	0	14
República Tcheca	10	0	10	10
França	0	0	0	4

Explicação

- Dimensões:
  - `Country`
- Medidas:
  - `Sum(Amount)`
    Soma `Amount` para a seleção atual. Observe que apenas `Canada` e `Czech Republic` têm valores diferentes de zero.
  - `Sum({<Country*={Canada}>}Amount)`
    Soma `Amount` para a seleção atual, cruzada com o requisito de que `Country` seja `Canada`. Se `Canada` não fizer parte da seleção do usuário, a expressão de conjunto retornará um conjunto vazio, e a coluna terá `0` em todas as linhas.
  - `Sum({<Country-={Canada}>}Amount)`
    Soma `Amount` para a seleção atual, mas primeiro exclui `Canada` da seleção `Country`. Se `Canada` não fizer parte da seleção do usuário, a expressão de conjunto não mudará nenhum número.
  - `Sum({<Country+={France}>}Amount)`
    Soma `Amount` para a seleção atual, mas primeiro adiciona `France` à seleção `Country`. Se `France` já fizer parte da seleção do usuário, a expressão de conjunto não mudará nenhum número.

Modificadores de conjunto com operadores de conjunto implícitos



Exemplos	Resultados
sum({$<Product += {OurProduct1, OurProduct2} >} Sales)	Retorna as vendas da seleção atual, mas usando uma união implícita para adicionar os produtos 'OurProduct1' e 'OurProduct2' à lista de produtos selecionados.
sum({$<Year += {“20*”,1997} – {2000} >} Sales)	Retorna as vendas da seleção atual, mas usando uma união implícita para adicionar um número de anos à seleção: 1997 e todos os anos que começam com "20" – exceto o ano 2000. Observe que se 2000 estivesse incluído na seleção atual, ele continuaria incluído após a modificação. Igual a `<Year=Year + ({“20*”,1997}-{2000})>`.

Exemplos	Resultados
sum({$<Product *= {OurProduct1} >} Sales)	Retorna as vendas da seleção atual, mas somente para a inserção dos produtos atualmente selecionados e do produto OurProduct1.

Modificadores de conjunto usando funções de conjunto

Às vezes, você precisa definir um conjunto de valores de campo usando uma definição de conjunto aninhada. Por exemplo, você pode querer selecionar todos os clientes que compraram um produto específico, sem selecionar o produto.

Nesses casos, use as funções do conjunto de elementos `P()` e `E()`. Elas retornam os conjuntos de elementos de valores possíveis e valores excluídos de um campo, respectivamente. Nos colchetes, você pode especificar o campo em questão e uma expressão de conjunto que define o escopo. Por exemplo:

```
P({1<Year = {2021}>} Customer)
```

Isso retornará o conjunto de clientes que realizaram transações em 2021. Você pode então usar isso em um modificador de conjunto. Por exemplo:

```
Sum({<Customer = P({1<Year = {2021}>} Customer)>} Amount)
```

Essa expressão de conjunto selecionará esses clientes, mas não restringirá a seleção a 2021.

Essas funções não podem ser usadas em outras expressões.

Além disso, apenas conjuntos naturais podem ser usados dentro das funções de conjunto de elementos. Ou seja, um conjunto de registos que podem ser definidas por uma seleção simples.

Por exemplo, o conjunto fornecido por `{1-$}` não pode ser sempre definido por meio de uma seleção e, portanto, não é um conjunto natural. O uso dessas funções em conjuntos não naturais retornará resultados inesperados.

Exemplos: Expressões de gráfico para modificadores de conjunto usando funções de conjunto

Exemplos - expressões de gráfico

Script de carregamento

Carregue os seguintes dados como um carregamento inline no editor de carregamento de dados para criar os exemplos de expressão de gráfico abaixo.

```
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];
```

Expressões de gráfico

Crie uma tabela em uma pasta do Qlik Sense com as seguintes expressões de gráfico.

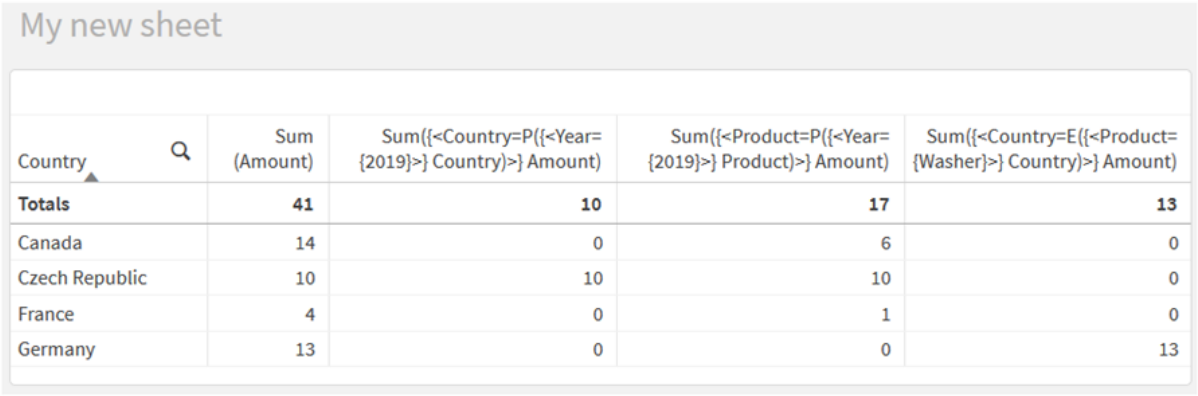
Tabela - Modificadores de conjunto usando funções de conjunto

Country	Sum (Amount)	Sum({<Country=P ({<Year= {2019}>}Country)>} Amount)	Sum({<Product=P ({<Year= {2019}>}Product)>} Amount)	Sum({<Country=E ({<Product= {Washer}>}Country)>} Amount)
Totais	**41**	**10**	**17**	**13**
Canadá	14	0	6	0
República Tcheca	10	10	10	0
França	4	0	1	0
Alemanha	13	0	0	13

Explicação

- Dimensões:
  - `Country`
- Medidas:
  - `Sum(Amount)`
    Soma `Amount`, sem expressão de conjunto.
  - `Sum({<Country=P({<Year={2019}>} Country)>} Amount)`
    Soma `Amount` para os países associados ao ano 2019. No entanto, não limitará o cálculo a 2019.
  - `Sum({<Product=P({<Year={2019}>} Product)>} Amount)`
    Soma `Amount` para os produtos que estão associados ao ano 2019. No entanto, não limitará o cálculo a 2019.
  - `Sum({<Country=E({<Product={Washer}>} Country)>} Amount)`
    Soma `Amount` para os países que não estão associados ao produto `Washer`.

Modificadores de conjunto usando funções de conjunto



Country	Sum (Amount)	Sum({<Country=P({<Year={2019}>} Country)>} Amount)	Sum({<Product=P({<Year={2019}>} Product)>} Amount)	Sum({<Country=E({<Product={Washer}>} Country)>} Amount)
Totals	**41**	**10**	**17**	**13**
Canada	14	0	6	0
Czech Republic	10	10	10	0
France	4	0	1	0
Germany	13	0	0	13

Exemplos	**Resultados**
sum({$<Customer = P({1<Product= {'Shoe'}>} Customer)>} Sales)	Retorna as vendas da seleção atual, mas somente os clientes que já compraram o produto 'Shoe'. Aqui, a função de elemento P( ) retorna uma lista de clientes possíveis; aqueles decorrentes da seleção 'Shoe' no campo Product.
sum({$<Customer = P({1<Product= {'Shoe'}>})>} Sales)	O mesmo que o anterior. Se o campo na função de elemento for omitido, a função retornará os valores possíveis do campo especificado na atribuição externa.
sum({$<Customer = P({1<Product= {'Shoe'}>} Supplier)>} Sales)	Retorna as vendas da seleção atual, mas somente os clientes que já forneceram o produto 'Shoe', ou seja, o cliente também é um fornecedor. Aqui, a função de elemento P( ) retorna uma lista de fornecedores possíveis; aqueles decorrentes da seleção 'Shoe' no campo Product. Assim, a lista de fornecedores é usada como uma seleção no campo Customer.
sum({$<Customer = E({1<Product= {'Shoe'}>})>} Sales)	Retorna as vendas da seleção atual, mas somente os clientes que nunca compraram o produto 'Shoe'. Aqui, a função de elemento E( ) retorna a lista de clientes excluídos; aqueles excluídos pela seleção 'Shoe' no campo Product.

Expressões de conjunto internas e externas

Expressões de conjunto podem ser usadas dentro e fora das funções de agregação e são colocadas entre colchetes.

Quando você usa uma expressão de conjunto dentro de uma função de agregação, ela pode ter a seguinte aparência:

Exemplo: Expressão de conjunto interna

```
Sum( {$<Year={2021}>} Sales )
```

Use uma expressão de conjunto fora da função de agregação se você tiver expressões com várias agregações e quiser evitar escrever a mesma expressão de conjunto em cada função de agregação.

Se você usar uma expressão de conjunto externa, ela deverá ser colocada no início do escopo.

Exemplo: Expressão de conjunto externa

```
{<Year={2021}>} Sum(Sales) / Count(distinct Customer)
```

Se você usar uma expressão de conjunto fora da função de agregação, também poderá aplicá-la em medidas mestre existentes.

Exemplo: Expressão de conjunto externa aplicada à medida mestre

```
{<Year={2021}>} [Master Measure]
```

Uma expressão de conjunto usada fora das funções de agregação afeta toda a expressão, a menos que esteja entre colchetes, que então definem o escopo. No exemplo de escopo léxico abaixo, a expressão de conjunto apenas é aplicada à agregação dentro dos colchetes.

Exemplo: Definição do escopo lexical

```
( {<Year={2021}>} Sum(Amount) / Count(distinct Customer) ) – Avg(CustomerSales)
```

Regras

Escopo lexical

A expressão de conjunto afeta a expressão inteira, a menos que esteja entre colchetes. Nesse caso, os colchetes definem o escopo lexical.

Posição

A expressão de conjunto deve ser colocada no início do escopo lexical.

Contexto

O contexto é a seleção que é relevante para a expressão. Tradicionalmente, o contexto sempre foi o estado padrão da seleção atual. Porém, se um objeto estiver definido em um estado alternativo, o contexto será o estado alternativo da seleção atual.

Você também pode definir um contexto na forma de uma expressão de conjunto externa.

Herança

Expressões de conjunto internas têm precedência sobre expressões de conjunto externas. Se a expressão de conjunto interna contiver um identificador de conjunto, ela substituirá o contexto. Caso contrário, o contexto e a expressão de conjunto serão mesclados.

- `{$<SetExpression>}`: substitui a expressão de conjunto externa
- `{<SetExpression>}`: é mesclado com a expressão de conjunto externa

Atribuição de conjunto de elementos

A atribuição de conjunto de elementos determina como as duas seleções são mescladas. Se um sinal de igual normal for usado, a seleção na expressão do conjunto interno terá precedência. Caso contrário, o operador de conjunto implícito será usado.

- `{<Field={value}>}`: essa seleção interna substitui qualquer seleção externa em "`Field`".
- `{<Field+={value}>}`: essa seleção interna é mesclada com a seleção externa em "`Field`", usando o operador de união.
- `{<Field*={value}>}`: essa seleção interna é mesclada com a seleção externa em "`Field`", usando o operador de interseção.

Herança em várias etapas

A herança pode ocorrer em várias etapas. Exemplos:

- Seleção atual → `Sum(Amount)`
  A função de agregação usará o contexto, que aqui é a seleção atual.
- Seleção atual → `{<Set1>} Sum(Amount)`
  `Set1` herdará da seleção atual, e o resultado será o contexto da função de agregação.
- Seleção atual → `{<Set1>} ({<Set2>} Sum(Amount))`
  `Set2` herdará de `Set1`, que, por sua vez, herdará da seleção atual, e o resultado será o contexto da função de agregação.

A função Aggr()

A função `Aggr()` cria uma agregação aninhada que tem duas agregações independentes. No exemplo abaixo, um `Count()` é calculado para cada valor de `Dim`, e a matriz resultante é agregada usando a função `Sum()`.

Exemplo:

```
Sum(Aggr(Count(X),Dim))
```

`Count()` é a agregação interna e `Sum()` é a agregação externa.

- A agregação interna não herda nenhum contexto da agregação externa.
- A agregação interna herda o contexto da função `Aggr()`, que pode conter uma expressão de conjunto.
- Tanto a função `Aggr()` quanto a função de agregação externa herdam o contexto de uma expressão de conjunto externa.

Tutorial - Criando uma expressão de conjunto

Você pode construir expressões de conjunto no Qlik Sense para oferecer suporte a análises de dados. Nesse contexto, a análise é frequentemente chamada de análise de conjunto. A análise de conjunto oferece uma maneira de definir um escopo diferente do conjunto de registros definido pela seleção atual em um aplicativo.

O que você aprenderá

Este tutorial fornece as expressões de dados e gráficos para criar expressões de conjunto usando modificadores de conjunto, identificadores e operadores.

Quem deve concluir este tutorial

Este tutorial é para desenvolvedores de aplicativos familiarizados com o trabalho com o editor de scripts e expressões de gráfico.

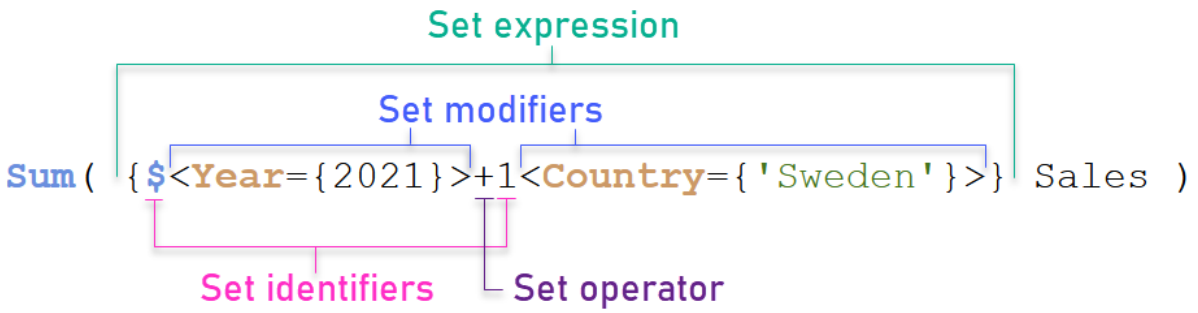
O que você deve fazer antes de começar

Uma alocação de acesso profissional do Qlik Sense Enterprise, que permite carregar dados e criar aplicativos.

Elementos em uma expressão de conjunto

Expressões de conjunto estão delimitadas em uma função de agregação, como `Sum()`, `Max()`, `Min()`, `Avg()` ou `Count()`. Expressões de conjunto são formadas a partir de blocos de construção conhecidos como elementos. Esses elementos são modificadores de conjuntos, identificadores e operadores.

Elementos em uma expressão de conjunto



A expressão de conjunto acima, por exemplo, é construída a partir da agregação `Sum(Sales)`. A expressão de conjunto é colocada entre colchetes externos: `{ }`

O primeiro operando da expressão é: `$<Year={2021}>`

Esse operando retorna as vendas do ano 2021 para a seleção atual. O modificador, `<Year={2021}>`, contém a seleção do ano de 2021. O identificador de conjunto `$` indica que a expressão de conjunto é baseada na seleção atual.

O segundo operando da expressão é: `1<Country={'Sweden'}>`

Esse operando retorna Sales para Sweden. O modificador, `<Country={'Sweden'}>`, contém a seleção do país Sweden. O identificador de conjunto `1` indica que as seleções feitas no aplicativo serão ignoradas.

Finalmente, o operador de conjunto `+` indica que a expressão retorna um conjunto que consiste nos registros que pertencem a qualquer um dos dois operandos de conjunto.

Tutorial - Criando uma expressão de conjunto

Conclua os procedimentos a seguir para criar as expressões de conjunto mostradas neste tutorial.

Criar um novo aplicativo e carregar os dados

Faça o seguinte:

1. Crie um novo aplicativo.
2. Clique em **Editor de script**. Como alternativa, clique em **Preparar** > **Editor de carregamento de dados** na barra de navegação.
3. Crie uma nova seção no **Editor de carregamento de dados**.
4. Copie os dados a seguir e cole-os na nova seção: *Dados do tutorial de expressão de conjunto (page 319)*
5. Clique em **Carregar dados**. Os dados são carregados como um carregamento inline.

Criar expressões de conjunto com modificadores

O modificador de conjunto consiste em um ou mais nomes de campo, cada um seguido por uma seleção que deve ser feita no campo. O modificador está entre sinais de maior e menor. Por exemplo, nesta expressão de conjunto:

```
Sum ( {<Year = {2015}>} Sales )
```

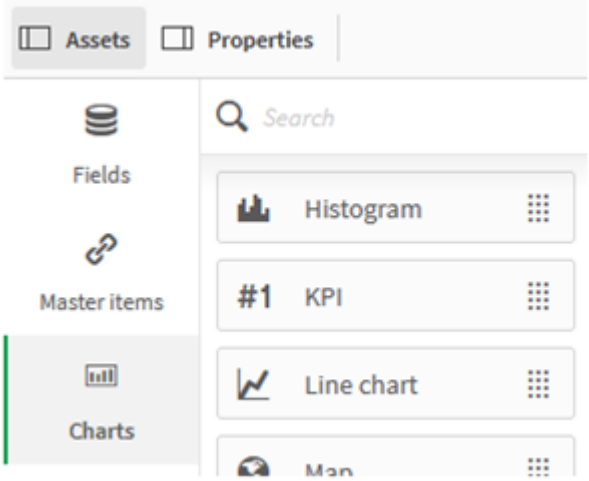
O modificador é:

```
<Year = {2015}>
```

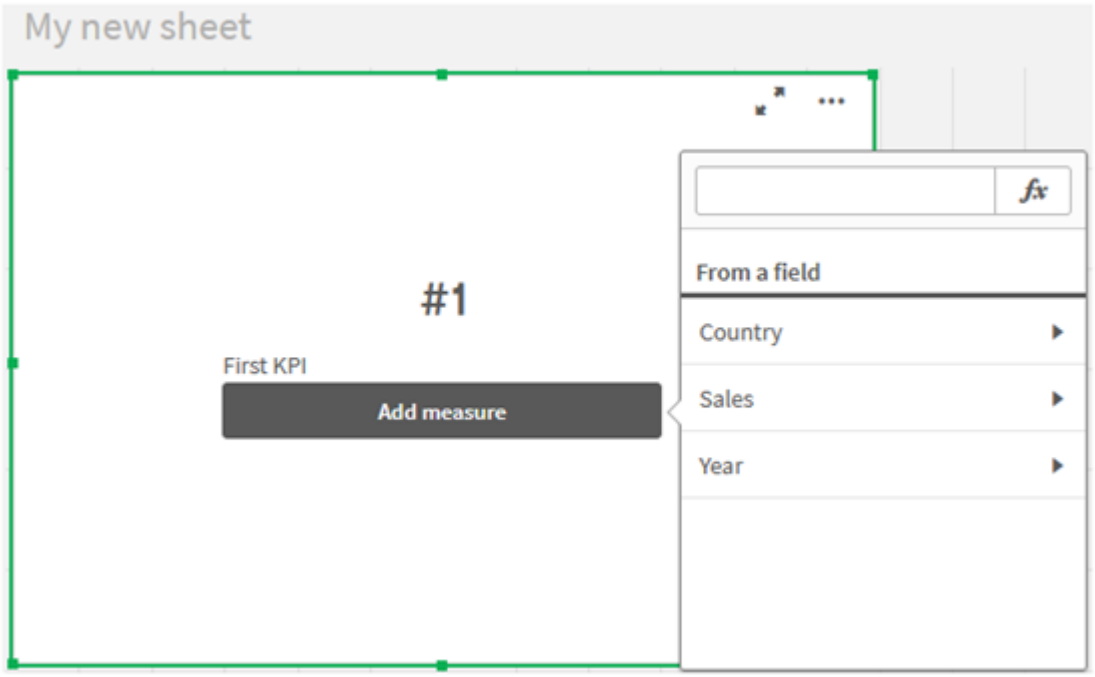
Esse modificador especifica que os dados do ano de 2015 serão selecionados. Os colchetes nos quais o modificador está incluído indicam uma expressão de conjunto.

Faça o seguinte:

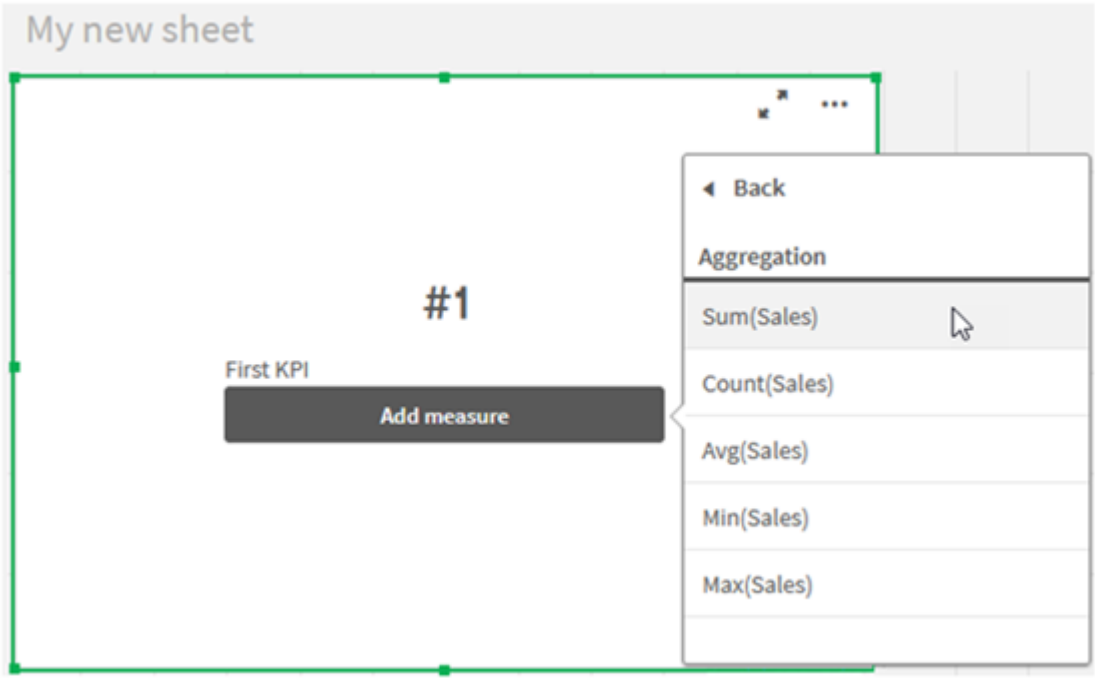
1. Em uma pasta, abra o painel **Ativos** na barra de navegação e clique em **Gráficos**.



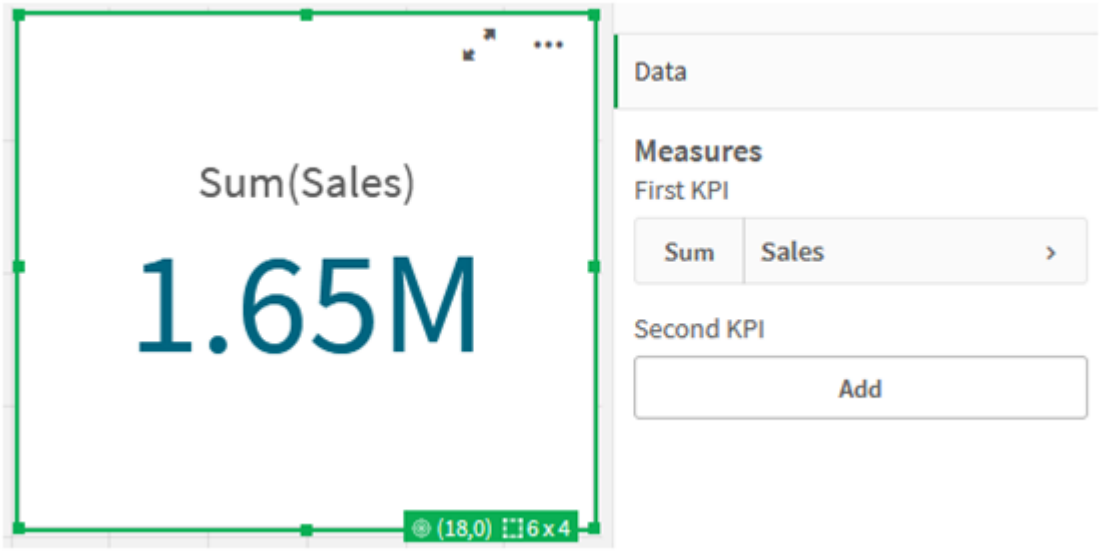
2. Arraste um **KPI** até a pasta e clique em **Adicionar medida**.



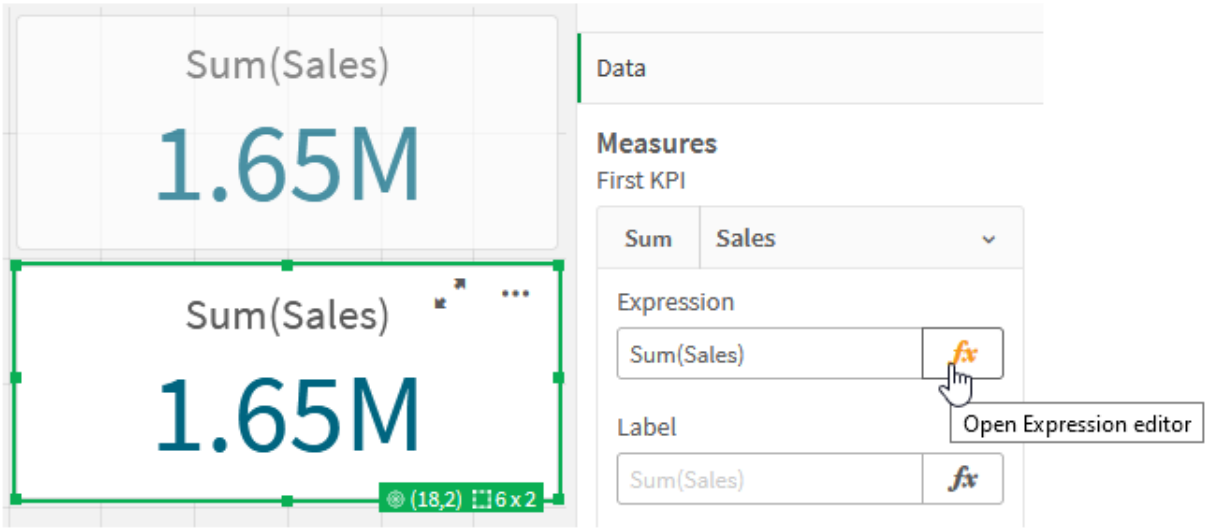
3. Clique em `Sales` e selecione `Sum(Sales)` para a agregação.



O KPI mostra a soma das vendas de todos os anos.

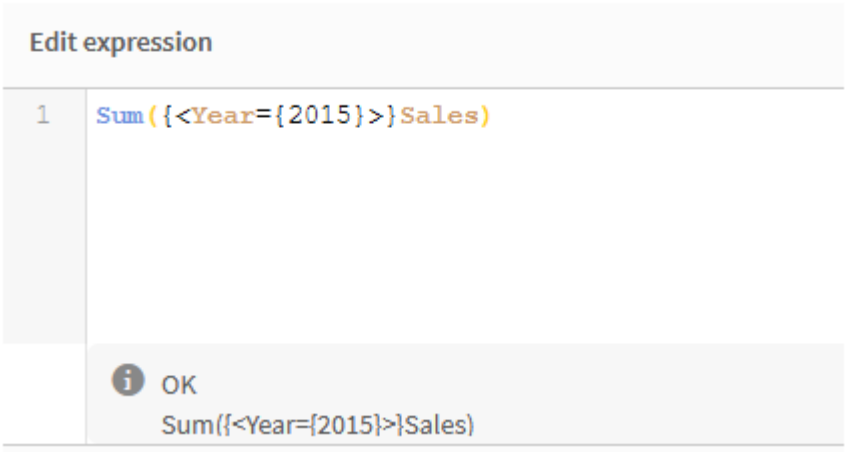


4. Copie e cole o KPI para criar um novo KPI.
5. Clique no novo KPI, clique em **Vendas** em **Medidas** e, em seguida, em **Abrir editor de expressões**.

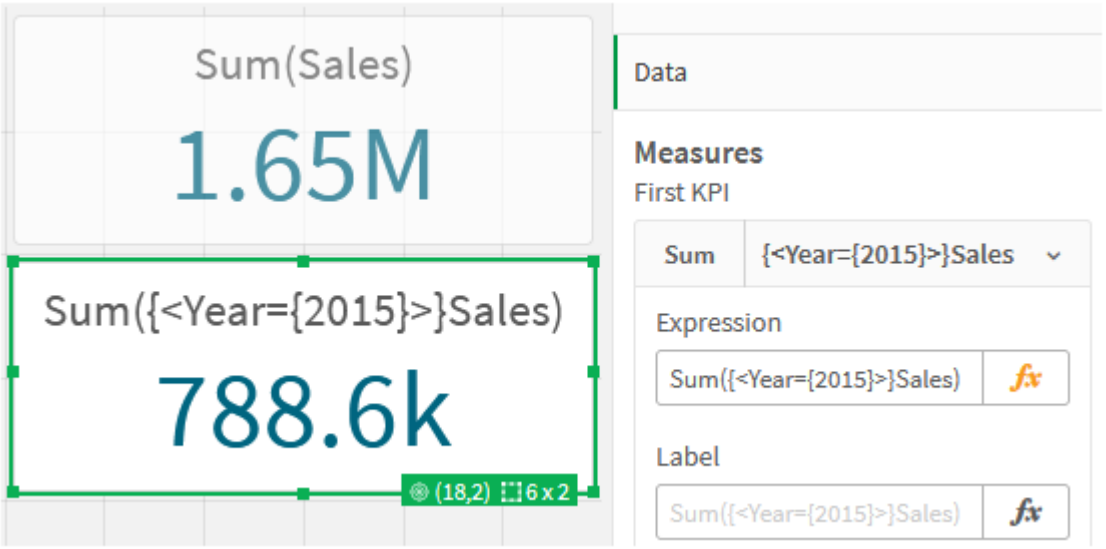


O editor de expressões é aberto com a agregação `Sum(Sales)`.

6. No editor de expressões, crie uma expressão para somar Sales somente para 2015:
   i. Adicione chaves para indicar uma expressão de conjunto: `Sum({}Sales)`
   i. Adicione colchetes angulares para indicar um modificador de conjunto: `Sum({<>}Sales)`
   ii. Nos colchetes angulares, adicione o campo a ser selecionado, neste caso o campo é `Year`, seguido por um sinal de igual. Em seguida, inclua 2015 em outro conjunto de chaves. O modificador de conjunto resultante é: `{<Year={2015}>}`.
   A expressão inteira é:
   `Sum({<Year={2015}>}Sales)`

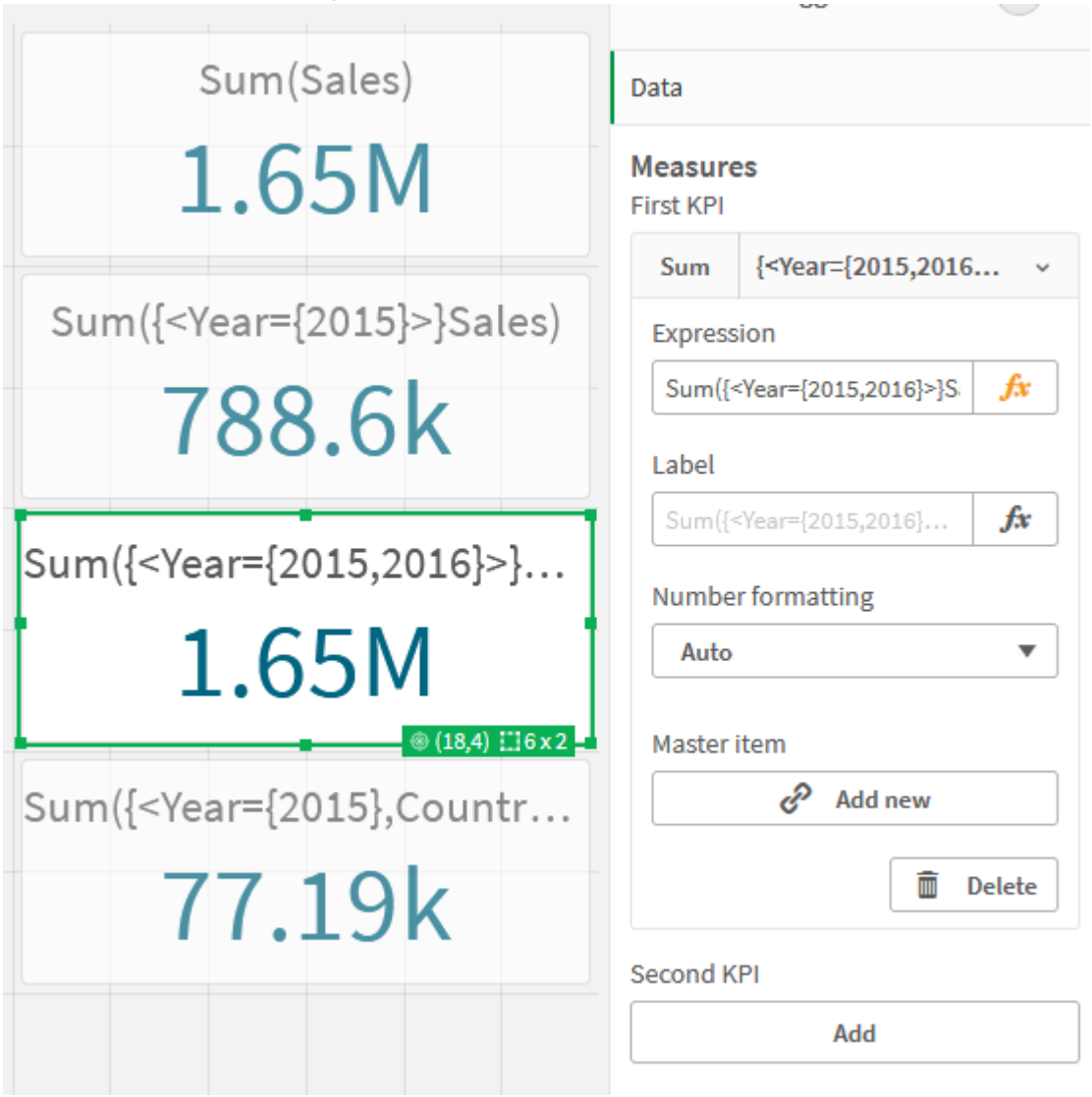


   iii. Clique em **Aplicar** para salvar a expressão e fechar o editor de expressões. A soma de Sales de 2015 é mostrada no KPI.



7. Crie mais dois KPIs com as seguintes expressões:
   `Sum({<Year={2015,2016}>}Sales)`
   O modificador acima é `<Year={2015,2016}>`. A expressão retornará a soma de Sales para 2015 e 2016.
   `Sum({<Year={2015},Country={'Germany'}>} Sales)`
   O modificador acima é `<Year={2015}, Country={'Germany'}>`. A expressão retornará a soma das vendas de Sales para 2015, em que 2015 faz interseção com Germany.

KPIs usando modificadores de conjunto



Adicionar identificadores de conjunto

As expressões de conjunto acima usarão as seleções atuais como base, porque um identificador não foi usado. Em seguida, adicione identificadores para especificar o comportamento quando seleções forem feitas.

Faça o seguinte:

Na sua pasta, crie ou copie as seguintes expressões de conjunto:

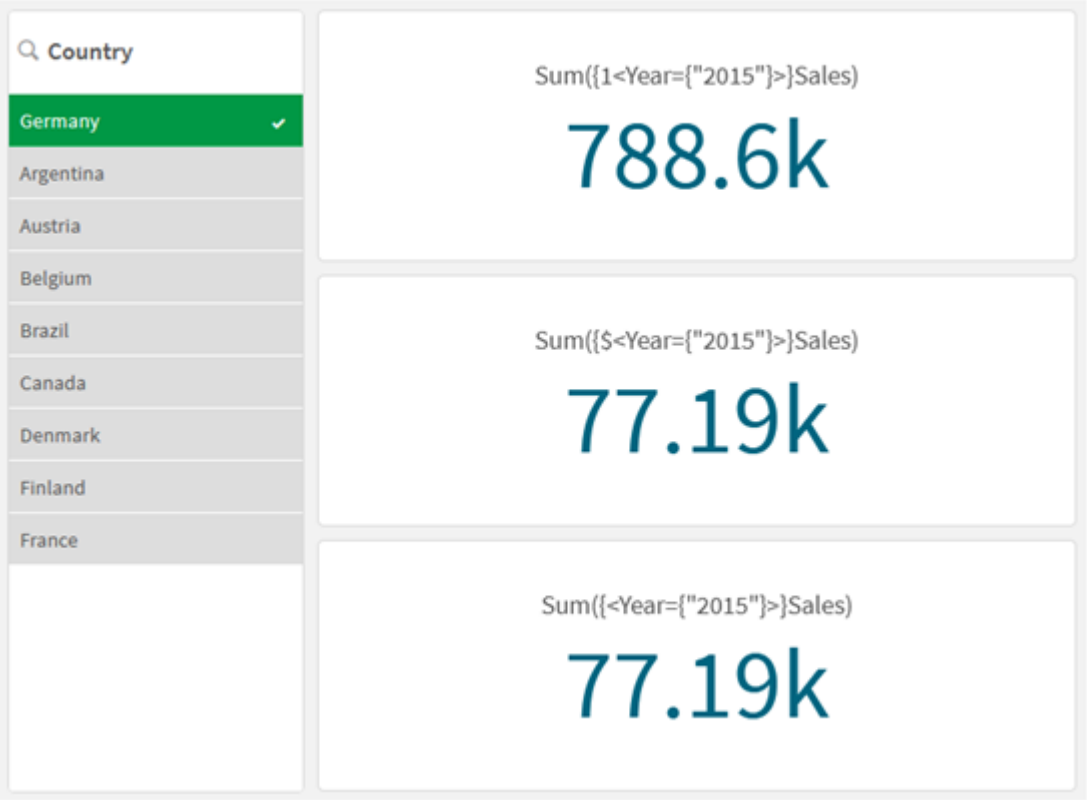
```
Sum({$<Year={"2015"}>}Sales
```

O identificador `$` baseará a expressão de conjunto nas seleções atuais feitas nos dados. Este também é o comportamento padrão quando um identificador não é usado.

```
Sum({1<Year={"2015"}>}Sales)
```

O identificador 1 fará com que a agregação de `Sum(Sales)` em 2015 ignore a seleção atual. O valor da agregação não mudará quando o usuário fizer outras seleções. Por exemplo, quando Germany é selecionado abaixo, o valor para a soma agregada de 2015 não muda.

KPIs usando modificadores e identificadores de conjunto



Adicionar operadores

Operadores de conjunto são usados para incluir, excluir ou cruzar conjuntos de dados. Todos os operadores usam conjuntos como operandos e retornam um conjunto como resultado.

Você pode usar operadores de conjunto em duas situações diferentes:

- Para executar uma operação de conjunto em identificadores de conjunto, representando conjuntos de registros em dados.
- Para executar uma operação de conjunto nos conjuntos de elementos, nos valores de campo ou dentro de um modificador de conjunto.

Faça o seguinte:

Na sua pasta, crie ou copie a seguinte expressão de conjunto:

```
Sum({$<Year={2015}>+1<Country={'Germany'}>}Sales)
```

O operador `(+)` de sinal de adição produz uma união dos conjuntos de dados para 2015 e Germany. Conforme explicado com identificadores de conjunto acima, o identificador `($)` de sinal de cifrão significa que as seleções atuais serão usadas para o primeiro operando, `<Year={2015}>`, e serão respeitadas. O identificador `1` significa que a seleção será ignorada para o segundo operando, `<Country={'Germany'}>`.

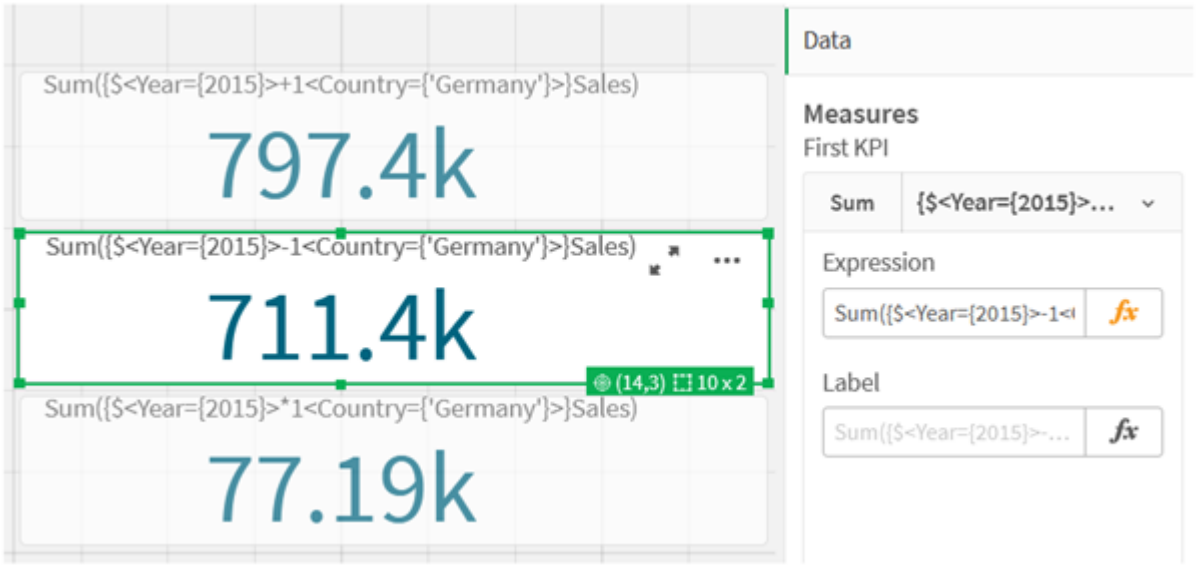
KPI usando o operador de sinal de mais (+)

Como alternativa, use um sinal de menos `(-)` para retornar um conjunto de dados que consiste nos registros que pertencem a 2015, mas não a Germany. Ou use um asterisco `(*)` para retornar um conjunto que consiste nos registros que pertencem a ambos os conjuntos.

```
Sum({$<Year={2015}>-1<Country={'Germany'}>}Sales)
```

```
Sum({$<Year={2015}>*1<Country={'Germany'}>}Sales)
```

KPIs usando operadores



Dados do tutorial de expressão de conjunto

Script de carregamento

Carregue os seguintes dados como um carregamento inline e, em seguida, crie as expressões de gráfico no 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
];
```

Sintaxe para expressões de conjunto

A sintaxe completa (sem incluir o uso opcional dos parênteses padrão para definir a precedência) é descrita usando o Formalismo Backus-Naur:

```
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 "
```

3.3 Sintaxe geral para expressões de gráfico

A seguinte estrutura de sintaxe geral pode ser usada para expressões de gráfico, com muitos parâmetros opcionais:

```
expression ::= ( constant | expressionname | operator1 expression | expression operator2
expression | function | aggregation function | (expression ) )
```

na qual:

constant é uma string (um texto, data ou hora) entre aspas simples retas ou um número. Constantes são escritas sem separador de milhar e com um ponto decimal como separador de decimal.

expressionname é o nome (rótulo) de outra expressão no mesmo gráfico.

operator1 é um operador unário (atuando em uma expressão, a da direita).

operator2 é um operador binário (atuando em duas expressões, uma de cada lado).

```
function ::= functionname ( parameters )
parameters ::= expression { , expression }
```
O número e os tipos de parâmetros não são arbitrários. Eles dependem da função utilizada.

```
aggregationfunction ::= aggregationfunctionname ( parameters2 )
parameters2 ::= aggrexpression { , aggrexpression }
```
O número e os tipos de parâmetros não são arbitrários. Eles dependem da função utilizada.

3.4 Sintaxe geral para agregações

A seguinte estrutura de sintaxe geral pode ser usada para agregações, com muitos parâmetros opcionais:

```
aggrexpression ::= ( fieldref | operator1 aggrexpression | aggrexpression operator2
aggrexpression | functioninaggr | ( aggrexpression ) )
```
fieldref é um nome de campo.

```
functionaggr ::= functionname ( parameters2 )
```
Assim, expressões e funções podem ser aninhadas livremente, desde que um **fieldref** venha sempre dentro de exatamente uma função de agregação e, desde que a expressão retorne um valor interpretável, o Qlik Sense não apresentará mensagens de erro.

4 Operadores

Esta seção descreve os operadores que podem ser usados no Qlik Sense. Existem dois tipos de operadores:

- Operadores unários (usam apenas um operando)
- Operadores binários (usam dois operandos)

A maior parte dos operadores é binária.

Podem ser definidos os seguintes operadores:

- Operadores de bit
- Operadores lógicos
- Operadores numéricos
- Operadores relacionais
- Operadores de caractere

4.1 Operadores de bit

Todos os operadores de bit convertem (truncam) os operandos em números inteiros assinados (32 bits) e retornam o resultado da mesma maneira. Todas as operações são realizadas bit por bit. Se um operando não puder ser interpretado como um número, a operação retornará NULL.

Operadores de bit

Operador	Nome completo	Descrição
bitnot	Bit inverso.	Operador unário. A operação retorna o inverso lógico do operando executado bit por bit. **Exemplo:** `bitnot 17` retorna -18.
bitand	Bit e.	A operação retorna o AND lógico dos operandos executados bit por bit. **Exemplo:** `17 bitand 7` retorna 1.
bitor	Bit ou.	A operação retorna o OR lógico dos operandos executados bit por bit. **Exemplo:** `17 bitor 7` retorna 23.

Operador	Nome completo	Descrição
bitxor	Bit ou exclusivo.	A operação retorna o OR exclusivo lógico dos operandos executados bit por bit. **Exemplo:** `17 bitxor 7` retorna 22.
>>	Bit deslocamento à direita.	A operação retorna o primeiro operando deslocado para a direita. O número de etapas é definido no segundo operando. **Exemplo:** `8 >> 2` retorna 2.
<<	Bit deslocamento à esquerda.	A operação retorna o primeiro operando deslocado para a esquerda. O número de etapas é definido no segundo operando. **Exemplo:** `8 << 2` retorna 32.

4.2 Operadores lógicos

Todos os operadores lógicos interpretam os operandos logicamente e retornam True (-1) ou False (0) como resultado.

Operadores lógicos

Operador	Descrição
not	Inverso lógico. Um dos operadores unários. A operação retorna o inverso lógico do operando.
and	And lógico. A operação retorna o and lógico dos operandos.
or	Or lógico. A operação retorna o or lógico dos operandos.
Xor	Or lógico exclusivo. A operação retorna o or lógico exclusivo dos operandos. Isto é, semelhante ao or lógico, mas o resultado será Falsese os dois operandos forem True.

4.3 Operadores numéricos

Todos os operadores numéricos usam os valores numéricos dos operandos e retornam um valor numérico como resultado.

Operadores numéricos

Operador	Descrição
+	Sinal para número positivo (operador unário) ou adição aritmética. A operação binária retorna a soma dos dois operandos.
-	Sinal para número negativo (operador unário) ou subtração aritmética. A operação unária retorna o operando multiplicado por -1 e, a binária, a diferença entre os dois operandos.
*	Multiplicação aritmética. A operação retorna o produto dos dois operandos.
/	Divisão aritmética. A operação retorna a razão entre os dois operandos.

4.4 Operadores relacionais

Todos os operadores relacionais comparam os valores dos operandos e retornam True (-1) ou False (0) como resultado. Todos os operadores relacionais são binários.

Operadores relacionais

Operador	Descrição
<	Menor que. Se os dois operandos puderem ser interpretados numericamente, será feita uma comparação numérica. A operação retorna o valor lógico da avaliação da comparação.
<=	Menor que ou igual a. Se os dois operandos puderem ser interpretados numericamente, será feita uma comparação numérica. A operação retorna o valor lógico da avaliação da comparação.
>	Maior que. Se os dois operandos puderem ser interpretados numericamente, será feita uma comparação numérica. A operação retorna o valor lógico da avaliação da comparação.
>=	Maior que ou igual a. Se os dois operandos puderem ser interpretados numericamente, será feita uma comparação numérica. A operação retorna o valor lógico da avaliação da comparação.
=	Igual a. Se os dois operandos puderem ser interpretados numericamente, será feita uma comparação numérica. A operação retorna o valor lógico da avaliação da comparação.
<>	Não equivalente a. Se os dois operandos puderem ser interpretados numericamente, será feita uma comparação numérica. A operação retorna o valor lógico da avaliação da comparação.

Operador	Descrição
precedes	Ao contrário do operador **<**, não é feita qualquer tentativa de interpretação numérica dos valores do argumento antes da comparação. A operação retornará verdadeiro se o valor à esquerda do operador tiver uma representação de texto que, na comparação da string, estiver antes da representação de texto do valor à direita. **Exemplo:** `'1 ' precedes ' 2'` retorna FALSE `' 1' precedes ' 2'`retorna TRUE pois o valor ASCII de um espaço (" ") é de menor valor do que o valor de um número ASCII. Compare com: `'1 ' < ' 2'` retorna TRUE `' 1' < ' 2'`retorna TRUE
follows	Ao contrário do operador **>**, não é feita qualquer tentativa de interpretação numérica dos valores do argumento antes da comparação. A operação retornará verdadeiro se o valor à esquerda do operador tiver uma representação de texto que, na comparação da string, vier depois da representação de texto do valor à direita. **Exemplo:** `' 2' follows '1'` retorna FALSE ' 2' follows ' 1'retorna TRUE pois o valor ASCII de um espaço (" ") é de menor valor do que o valor de um número ASCII. Compare com: `' 2' > ' 1'`retorna TRUE `' 2' > '1 '`retorna TRUE

4.5 Operadores de caractere

Existem dois operadores de string. Um deles utiliza os valores de string dos operandos e retorna uma string como resultado. O outro compara os operandos e retorna um valor booliano para indicar a correspondência.

&

Concatenação de string. A operação retorna uma string de texto que consiste em duas strings de operandos, uma após a outra.

Exemplo:

`'abc' & 'xyz'` retorna 'abcxyz'

like

Comparação da string com caracteres curinga. A operação retornará um booleano True (-1) se a string que se encontrar antes do operador corresponder à string que estiver após o operador. A segunda cadeia pode conter os caracteres curinga * (qualquer número de caracteres arbitrários) ou ? (um caractere arbitrário).

Exemplo:

`'abc' like 'a*'` retorna True (-1)

`'abcd' like 'a?c*'` retorna True (-1)

`'abc' like 'a??bc'` retorna False (0)

5 Funções de script e gráfico

Transforme e agregue dados usando funções em scripts de carregamento de dados e expressões de gráfico.

Muitas funções podem ser usadas da mesma maneira nos scripts de carga de dados e em expressões de gráfico, mas há algumas exceções:

- Algumas funções só podem ser usadas nos scripts de carregamento de dados, indicadas por – função de script.
- Algumas funções só podem ser usadas em expressões de gráfico, indicadas por – função de gráfico.
- Algumas funções podem ser usadas nos scripts de carga de dados e em expressões de gráfico, mas com diferenças nos parâmetros e na aplicação. Elas são descritas em tópicos separados indicados por – função de script ou função de gráfico.

5.1 Conexões analíticas para extensões do lado do servidor (SSE)

As funções habilitadas por conexões analíticas só estarão visíveis se tiver configurado as conexões analíticas e se o Qlik Sense for iniciado.

Você configura as conexões analíticas no QMC. Consulte o tópico "Criando uma conexão analítica" no guia Gerenciar sites do Qlik Sense.

No Qlik Sense Desktop, você configura as conexões analíticas editando o arquivo *Settings.ini*; consulte o tópico " Configurando conexões analíticas no Qlik Sense Desktop" no guia Qlik Sense Desktop.

5.2 Funções de agregação

A família de funções conhecidas como funções de agregação consiste em funções que recebem vários valores de campos como sua entrada e retornam um único resultado por grupo, onde o agrupamento é definido por uma dimensão de gráfico ou uma cláusula **group by** na instrução de script.

Funções de agregação incluem **Sum()**, **Count()**, **Min()**, **Max()**, e muito mais.

A maioria das funções de agregação pode ser usadas no script de carregamento de dados e em expressões de gráfico, mas a sintaxe é diferente.

Limitações:

O parâmetro da função de agregação não deve conter outras funções de agregação, a menos que essas agregações internas contenham o qualificador **TOTAL**. Para agregações aninhadas mais avançadas, use a função avançada **Aggr** junto com uma dimensão especificada.

Ao nomear uma entidade, evite atribuir o mesmo nome a mais de um campo, variável ou medida. Existe uma ordem estrita de precedência para resolver conflitos entre entidades com nomes idênticos. Essa ordem é refletida em quaisquer objetos ou contextos nos quais essas entidades são usadas. Esta ordem de

precedência é a seguinte:

- Dentro de uma agregação, um campo tem precedência sobre uma variável. Rótulos de medida não são relevantes em agregações e não são priorizados.
- Fora de uma agregação, um rótulo de medida tem precedência sobre uma variável que, por sua vez, tem precedência sobre um nome de campo.
- Além disso, fora de uma agregação, uma medida pode ser reutilizada ao se fazer referência ao seu rótulo, a menos que esse rótulo seja de fato um rótulo calculado. Nessa situação, a medida perde significado para reduzir o risco de autorreferência e, nesse caso, o nome sempre será interpretado primeiro como um rótulo de medida, segundo como um nome de campo e terceiro como um nome de variável.

Usando funções de agregação em um script de carga de dados

Funções de agregação só podem ser usadas em comandos **LOAD** e **SELECT**.

Usando funções de agregação em expressões do gráfico

O parâmetro da função de agregação não deve conter outras funções de agregação, a menos que essas agregações internas contenham o qualificador **TOTAL**. Para agregações aninhadas mais avançadas, use a função avançada **Aggr** junto com uma dimensão especificada.

Uma função de agregação agrega sobre um conjunto de registros possíveis definidos pela seleção. No entanto, um conjunto de registros alternativos pode ser definido usando uma expressão na análise de conjunto.

Como agregações são calculadas

Uma agregação percorre os registros de uma tabela específica, agregando os registros nela. Por exemplo, **Count**(<Field>) contará o número de registros na tabela em que <Field> reside. Se quiser agregar apenas os valores de campo distintos, você precisará usar a cláusula **distinct**, como **Count**(**distinct** <Field>).

Se a função de agregação contiver campos de tabelas diferentes, a função de agregação percorrerá os registros do produto cruzado das tabelas dos campos constituintes. Como isso deteriora o desempenho, essas agregações devem ser evitadas, particularmente quando você tem grandes quantidades de dados.

Agregação de campos-chave

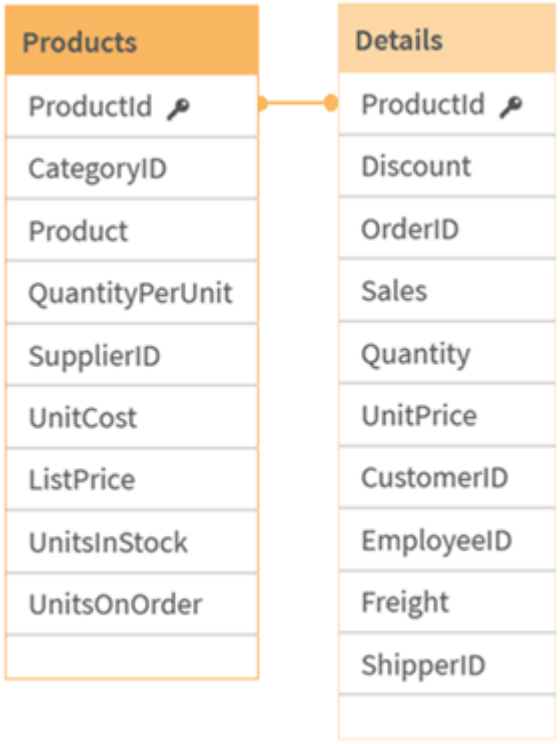
A maneira como as agregações são calculadas significa que você não pode agregar campos-chave porque não está claro qual tabela deve ser usada para a agregação. Por exemplo, se o campo <Key> vincula duas tabelas, não está claro se **Count**(<Key>) deve retornar o número de registros da primeira ou da segunda tabela.

No entanto, se você usar a cláusula **distinct**, a agregação estará bem definida e poderá ser calculada.

Portanto, se você usar um campo chave dentro de uma função de agregação sem a cláusula **distinct**, o Qlik Sense retornará um número que pode não fazer sentido. A solução é usar a cláusula **distinct** ou uma cópia da chave: uma cópia que resida em apenas uma tabela.

Por exemplo, nas seguintes tabelas, ProductID é a chave entre as tabelas.

Chave de ProductID entre as tabelas Produtos e Detalhes



Count(ProductID) pode ser contado na tabela Products (que tem apenas um registro por produto – ProductID é a chave primária) ou pode ser contado na tabela Details (que muito provavelmente tem vários registros por produto). Se você quiser contar o número de produtos distintos, deve usar o Count(distinct ProductID). Se você quiser contar o número de linhas em uma tabela específica, não deve usar a chave.

Funções básicas de agregação

Visão geral das funções básicas de agregação

As funções básicas de agregação são um grupo das funções de agregação mais comuns.

Cada função é descrita adicionalmente após a visão geral. Você também pode clicar no nome da função na sintaxe para acessar imediatamente os detalhes dessa função específica.

Funções de agregação básica no script de carga de dados

FirstSortedValue

FirstSortedValue() retorna o valor da expressão especificada em **value** que corresponde ao resultado de classificação do argumento **sort_weight**, por exemplo, o nome do produto com o menor preço de unidade. O valor nth na ordem de classificação pode ser especificado em **rank**. Se mais de um valor resultante compartilhar o mesmo **sort_weight** para o **rank**, especificado, a função retorna NULL. Os valores classificados são iterativos em número de registros, como definidos por uma cláusula **group by**, ou agregados em um conjunto de dados completo se nenhuma cláusula **group by** for definida.

```
FirstSortedValue ([ distinct ] expression, sort_weight [, rank ])
```

Max

Max() encontra o maior valor numérico dos dados agregados na expressão, conforme definido por uma cláusula **group by**. Especificando uma **rank** n, o enésimo valor mais alto pode ser encontrado.

```
Max ( expression[, rank])
```

Min

Min() retorna o menor valor numérico dos dados agregados na expressão, conforme definido por uma cláusula **group by**. Especificando uma **rank** n, o enésimo valor mais baixo pode ser encontrado.

```
Min ( expression[, rank])
```

Mode

Mode() retorna o valor que ocorre com mais frequência, o valor modal, dos dados agregados na expressão, conforme definido por uma cláusula **group by**. A função **Mode()** pode retornar valores numéricos além de valores de texto.

```
Mode (expression )
```

Only

Only() retorna um valor se houver um e somente um resultado possível a partir dos dados agregados. Se os registros contiverem apenas um valor, esse valor é retornado; caso contrário, NULL é retornado. Use a cláusula **group by** para avaliar múltiplos registros. A função **Only()** pode retornar valores numéricos e de texto.

```
Only (expression )
```

Sum

Sum() calcula o total dos valores agregados na expressão, conforme definido pela cláusula **group by**.

```
Sum ([distinct]expression)
```

Funções de agregação básica em expressões de gráfico

As funções de agregação de gráfico só podem ser usadas em campos de expressões de gráficos. A expressão do argumento de uma função de agregação não deve conter outra função de agregação.

FirstSortedValue

FirstSortedValue() retorna o valor da expressão especificada em **value** que corresponde ao resultado de classificação do argumento **sort_weight**, por exemplo, o nome do produto com o menor preço de unidade. O valor nth na ordem de classificação pode ser especificado em **rank**. Se mais de um valor resultante compartilhar o mesmo **sort_weight** para o **rank**, especificado, a função retorna NULL.

```
FirstSortedValue - função de gráfico([{SetExpression}] [DISTINCT] [TOTAL
[<fld {,fld}>]] value, sort_weight [,rank])
```

Max

Max() encontra o valor mais alto dos dados agregados. Especificando uma **rank** n, o enésimo valor mais alto pode ser encontrado.

```
Max - função de gráficoMax() encontra o valor mais alto dos dados agregados.
Especificando uma rank n, o enésimo valor mais alto pode ser encontrado.
Talvez você também queira consultar FirstSortedValue e rangemax, que têm uma
```

operação semelhante à função Max. Max([{SetExpression}] [TOTAL [<fld {,fld}>]] expr [,rank]) numérico ArgumentosArgumentoDescriçãoexprA expressão ou campo que contém os dados a serem medidos.rankO valor padrão de rank é 1, que corresponde ao valor mais alto. Se rank for especificada com o valor 2, o segundo valor mais alto é retornado. Se rank for 3, o terceiro valor mais alto é retornado, e assim por diante.SetExpressionPor padrão, a função de agregação agregará um conjunto de registros possíveis definidos pela seleção. Um conjunto de registros alternativos pode ser definido por uma expressão de análise. TOTALSe a palavra TOTAL ocorrer antes dos argumentos da função, o cálculo será feito sobre todos os valores possíveis, dadas as seleções atuais, e não apenas sobre os pertinentes ao valor dimensional atual, isto é, desconsiderando as dimensões do gráfico. Usando TOTAL [<fld {.fld}>], em que o qualificador TOTAL é seguido por uma lista de um ou mais nomes de campo como um subconjunto das variáveis de dimensão de gráfico, você cria um subconjunto dos valores possíveis
totais. DadosCustomerProductUnitSalesUnitPrice AstridaAA416AstridaAA1015AstridaBB99BetacabBB510BetacabCC220BetacabDD-25CanutilityAA815CanutilityCC-19Exemplos e resultadosExemplosResultadosMax (UnitSales)10, porque esse é o valor mais alto em UnitSales.O valor de um pedido é calculado a partir do número de unidades vendidas em (UnitSales), multiplicado pelo preço unitário.Max(UnitSales*UnitPrice)150, porque esse é o valor mais alto do resultado de calcular todos os valores possíveis de (UnitSales)*(UnitPrice).Max(UnitSales, 2)9, que é o segundo maior valor.Max (TOTAL UnitSales)10, pois o qualificador TOTAL significa que o maior valor possível é encontrado, desconsiderando as dimensões do gráfico. Para um gráfico com Customer como a dimensão, o qualificador TOTAL garantirá que o valor máximo em todo o conjunto de dados seja retornado, em vez da UnitSales máxima para cada cliente.Faça a seleção Customer B.Max({1}
TOTAL UnitSales)10, independentemente da seleção feita, pois a expressão Set Analysis {1} define o conjunto de registros a ser avaliado como ALL, independente da seleção feita.Dados usados nos exemplos:ProductData:LOAD * inline
[Customer|Product|UnitSales|UnitPriceAstrida|AA|4|16Astrida|AA|10|15Astrida|BB|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() encontra o valor mais baixo dos dados agregados. Especificando uma **rank** n, o enésimo valor mais baixo pode ser encontrado.

Min - função de gráfico([{SetExpression}] [DISTINCT] [TOTAL [<fld {,fld}>]] expr [,rank]**)**

Mode

Mode() encontra o valor mais geralmente ocorrido, o valor do modo, nos dados agregados. A função **Mode()** pode processar valores de texto, além de valores numéricos.

```
Mode - função de gráfico ({[SetExpression] [TOTAL [<fld {,fld}>]]} expr)
```

Only

Only() retorna um valor se houver um e somente um resultado possível a partir dos dados agregados. Por exemplo, procurar o único produto em que o preço unitário =9 retornará NULL se mais de um produto tiver um preço unitário de 9.

```
Only - função de gráfico([{SetExpression}] [DISTINCT] [TOTAL [<fld {,fld}>]]
expr)
```

Sum

Sum() calcula o total dos valores dados pela expressão ou campo em todos os dados agregados.

```
Sum - função de gráfico([{SetExpression}] [DISTINCT] [TOTAL [<fld {,fld}>]]
expr])
```

FirstSortedValue

FirstSortedValue() retorna o valor da expressão especificada em **value** que corresponde ao resultado de classificação do argumento **sort_weight**, por exemplo, o nome do produto com o menor preço de unidade. O valor nth na ordem de classificação pode ser especificado em **rank**. Se mais de um valor resultante compartilhar o mesmo **sort_weight** para o **rank**, especificado, a função retorna NULL. Os valores classificados são iterativos em número de registros, como definidos por uma cláusula **group by**, ou agregados em um conjunto de dados completo se nenhuma cláusula **group by** for definida.

Sintaxe:

```
FirstSortedValue ([ distinct ] value, sort-weight [, rank ])
```

Tipo de dados de retorno: dual

Argumentos:

Argumentos

Argumento	Descrição
value Expression	A função encontra o valor da expressão **value** que corresponde ao resultado de classificação **sort_weight**.
sort-weight Expression	A expressão que contém os dados que serão classificados. O primeiro valor (mais baixo) de **sort_weight** é encontrado, a partir do qual o valor correspondente da expressão **value** é determinado. Ao colocar um sinal de menos na frente de **sort_weight**, a função retorna o último (mais alto) valor classificado em seu lugar.
rank Expression	Se for especificado um **rank** "n" maior que 1, será obtido o enésimo valor classificado.
distinct	Se a palavra **DISTINCT** aparecer antes dos argumentos de função, as duplicatas resultantes da avaliação dos argumentos de função serão ignoradas.

Exemplos e resultados:

Adicione o script de exemplo ao seu aplicativo e execute-o. Em seguida, adicione pelo menos os campos listados na coluna de resultados a uma pasta para ver o resultado.

Para obter a mesma aparência que na coluna de resultados abaixo, no painel de propriedades, em Classificação, alterne de Automática para Personalizada e desmarque a classificação numérica e alfabética.

Exemplos de scripts

Exemplo	**Resultado**

`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`
	A função classifica UnitSales do menor para o maior, procurando o valor de Customer com o menor valor de UnitSales, o menor pedido.
	
	Como CC corresponde ao menor valor de pedido (valor de UnitSales=2) para o cliente Astrida. AA corresponde ao menor pedido (4) para o cliente Betacab, AA corresponde ao menor pedido (8) para o clienteCanutility, e DD corresponde ao menor pedido (10) para o clienteDivadip..

Dado que a tabela **Temp** é carregada como no exemplo anterior:
 	
`LOAD Customer,FirstSortedValue(Product, -UnitSales)`
`as MyProductWithLargestOrderByCustomer Resident`
`Temp Group By Customer;`
	`Customer`
	`MyProductWithLargestOrderByCustomer`
	`Astrida AA`
	`Betacab DD`
	`Canutility CC`
	`Divadip -`
	Um sinal de menor precede o argumento sort_weight, assim a função classifica primeiro o maior.
	
	Porque AA corresponde ao maior pedido (valor de UnitSales:18) do cliente Astrida, DD corresponde ao maior pedido (12) do cliente Betacab e CC corresponde ao maior pedido (13) do cliente Canutility. Há dois valores idênticos no maior pedido (16) para o cliente Divadip, portanto, isso produz um resultado nulo.

Exemplo	Resultado

Dado que a tabela **Temp** é carregada como no exemplo anterior:
`LOAD Customer,FirstSortedValue(distinct Product, -UnitSales) as MyProductWithSmallestOrderByCustomer Resident Temp Group By Customer;`
	`Customer MyProductWithLargestOrderByCustomer`
	`Astrida AA`
	`Betacab DD`
	`Canutility CC`
	`Divadip AA`
	É o mesmo que o exemplo anterior, exceto que o qualificador distinct é usado. Isso causa resultado duplicado para Divadip ser desconsiderado, permitindo que seja retornado um valor não nulo.

FirstSortedValue - função de gráfico

FirstSortedValue() retorna o valor da expressão especificada em **value** que corresponde ao resultado de classificação do argumento **sort_weight**, por exemplo, o nome do produto com o menor preço de unidade. O valor nth na ordem de classificação pode ser especificado em **rank**. Se mais de um valor resultante compartilhar o mesmo **sort_weight** para o **rank**, especificado, a função retorna NULL.

Sintaxe:

```
FirstSortedValue([{SetExpression}] [DISTINCT] [TOTAL [<fld {,fld}>]] value, sort_weight [,rank])
```

Tipo de dados de retorno: dual

Argumentos:

Argumentos

Argumento	Descrição
value	Campo de saída. A função encontra o valor da expressão **value** que corresponde ao resultado de classificação **sort_weight**.
sort_weight	Campo de entrada. A expressão que contém os dados que serão classificados. O primeiro valor (mais baixo) de **sort_weight** é encontrado, a partir do qual o valor correspondente da expressão **value** é determinado. Ao colocar um sinal de menos na frente de **sort_weight**, a função retorna o último (mais alto) valor classificado em seu lugar.
rank	Se for especificado um **rank** "n" maior que 1, será obtido o enésimo valor classificado.
SetExpression	Por padrão, a função de agregação agregará um conjunto de registros possíveis definidos pela seleção. Um conjunto de registros alternativos pode ser definido por uma expressão de análise.
DISTINCT	Se a palavra **DISTINCT** aparecer antes dos argumentos de função, as duplicatas resultantes da avaliação dos argumentos de função serão ignoradas.

Argumento	Descrição
TOTAL	Se a palavra **TOTAL** ocorrer antes dos argumentos da função, o cálculo será feito sobre todos os valores possíveis, dadas as seleções atuais, e não apenas sobre os pertinentes ao valor dimensional atual, isto é, desconsiderando as dimensões do gráfico. Usando **TOTAL [<fld {.fld}>]**, em que o qualificador **TOTAL** é seguido por uma lista de um ou mais nomes de campo como um subconjunto das variáveis de dimensão de gráfico, você cria um subconjunto dos valores possíveis totais.

Exemplos e resultados:

Dados

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

Exemplos e resultados

Exemplo	Resultado
`firstsortedvalue (Product, UnitPrice)`	BB, que é o `Product`com o menor `UnitPrice`(9).
`firstsortedvalue (Product, UnitPrice, 2)`	BB, que é o `Product`com o segundo menor `UnitPrice`(10).
`firstsortedvalue (Customer, -UnitPrice, 2)`	Betacab, que é o `Customer`com o `Product` com o segundo maior `UnitPrice`(20).
`firstsortedvalue (Customer, UnitPrice, 3)`	NULL, porque existem dois valores de `Customer` (Astrida eCanutility) com o mesmo rank (terceiro menor) `UnitPrice`(15). Use o qualificador distinct para se certificar de que resultados nulos não ocorrerão.
`firstsortedvalue (Customer, -UnitPrice*UnitSales, 2)`	Canutility, que é o `Customer` com o segundo maior valor de ordem de venda `UnitPrice` multiplicado por `UnitSales` (120).

Dados usados nos exemplos:

```
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() encontra o maior valor numérico dos dados agregados na expressão, conforme definido por uma cláusula **group by**. Especificando uma **rank** n, o enésimo valor mais alto pode ser encontrado.

Sintaxe:

```
Max ( expr [, rank])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
expr Expression	A expressão ou campo que contém os dados a serem medidos.
rank Expression	O valor padrão de **rank** é 1, que corresponde ao valor mais alto. Se **rank** for especificada com o valor 2, o segundo valor mais alto é retornado. Se **rank** for 3, o terceiro valor mais alto é retornado, e assim por diante.

Exemplos e resultados:

Adicione o script de exemplo ao seu aplicativo e execute-o. Em seguida, adicione pelo menos os campos listados na coluna de resultados a uma pasta para ver o resultado.

Para obter a mesma aparência que na coluna de resultados abaixo, no painel de propriedades, em Classificação, alterne de Automática para Personalizada e desmarque a classificação numérica e alfabética.

Exemplo:

```
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;
```

Tabela resultante

Customer	MyMax
Astrida	18
Betacab	5
Canutility	8

Exemplo:

Dado que a tabela **Temp** é carregada como no exemplo anterior:

```
LOAD Customer, Max(UnitSales,2) as MyMaxRank2 Resident Temp Group By Customer;
```

Tabela resultante

Customer	MyMaxRank2
Astrida	10
Betacab	4
Canutility	-

Max - função de gráfico

Max() encontra o valor mais alto dos dados agregados. Especificando uma **rank** n, o enésimo valor mais alto pode ser encontrado.

*Talvez você também queira consultar **FirstSortedValue** e **rangemax**, que têm uma operação semelhante à função **Max**.*

Sintaxe:

```
Max([{SetExpression}] [TOTAL [<fld {,fld}>]] expr [,rank])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
expr	A expressão ou campo que contém os dados a serem medidos.

Argumento	Descrição
rank	O valor padrão de **rank** é 1, que corresponde ao valor mais alto. Se **rank** for especificada com o valor 2, o segundo valor mais alto é retornado. Se **rank** for 3, o terceiro valor mais alto é retornado, e assim por diante.
SetExpression	Por padrão, a função de agregação agregará um conjunto de registros possíveis definidos pela seleção. Um conjunto de registros alternativos pode ser definido por uma expressão de análise.
TOTAL	Se a palavra **TOTAL** ocorrer antes dos argumentos da função, o cálculo será feito sobre todos os valores possíveis, dadas as seleções atuais, e não apenas sobre os pertinentes ao valor dimensional atual, isto é, desconsiderando as dimensões do gráfico. Usando **TOTAL [<fld {.fld}>]**, em que o qualificador **TOTAL** é seguido por uma lista de um ou mais nomes de campo como um subconjunto das variáveis de dimensão de gráfico, você cria um subconjunto dos valores possíveis totais.

Exemplos e resultados:

Dados

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

Exemplos e resultados

Exemplos	Resultados
`Max(UnitSales)`	10, porque esse é o valor mais alto em `UnitSales`.

Exemplos	Resultados
O valor de um pedido é calculado a partir do número de unidades vendidas em (UnitSales), multiplicado pelo preço unitário. `Max(UnitSales*UnitPrice)`	150, porque esse é o valor mais alto do resultado de calcular todos os valores possíveis de (UnitSales)*(UnitPrice).
`Max(UnitSales, 2)`	9, que é o segundo maior valor.
`Max(TOTAL UnitSales)`	10, pois o qualificador TOTAL significa que o maior valor possível é encontrado, desconsiderando as dimensões do gráfico. Para um gráfico com Customer como a dimensão, o qualificador TOTAL garantirá que o valor máximo em todo o conjunto de dados seja retornado, em vez da UnitSales máxima para cada cliente.
Faça a seleção `Customer B`. `Max({1} TOTAL UnitSales)`	10, independentemente da seleção feita, pois a expressão Set Analysis {1} define o conjunto de registros a ser avaliado como ALL, independente da seleção feita.

Dados usados nos exemplos:

```
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 '|');
```

Consulte também:

- *FirstSortedValue - função de gráfico (page 334)*
- *RangeMax (page 1351)*

Min

Min() retorna o menor valor numérico dos dados agregados na expressão, conforme definido por uma cláusula **group by**. Especificando uma **rank** n, o enésimo valor mais baixo pode ser encontrado.

Sintaxe:

```
Min ( expr [, rank])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
expr Expression	A expressão ou campo que contém os dados a serem medidos.
rank Expression	O valor padrão de **rank** é 1, que corresponde ao valor mais baixo. Se **rank** for especificado com 2, o segundo valor mais baixo será retornado. Se **rank** for 3, o terceiro valor mais baixo será retornado, e assim por diante.

Exemplos e resultados:

Adicione o script de exemplo ao seu aplicativo e execute-o. Em seguida, adicione pelo menos os campos listados na coluna de resultados a uma pasta para ver o resultado.

Para obter a mesma aparência que na coluna de resultados abaixo, no painel de propriedades, em Classificação, alterne de Automática para Personalizada e desmarque a classificação numérica e alfabética.

Exemplo:

```
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;
```

Tabela resultante

Customer	MyMin
Astrida	2
Betacab	4
Canutility	8

Exemplo:

Dado que a tabela **Temp** é carregada como no exemplo anterior:

```
LOAD Customer, Min(UnitSales,2) as MyMinRank2 Resident Temp Group By Customer;
```

Tabela resultante

Customer	MyMinRank2
Astrida	9
Betacab	5
Canutility	-

Min - função de gráfico

Min() encontra o valor mais baixo dos dados agregados. Especificando uma **rank** n, o enésimo valor mais baixo pode ser encontrado.

*Talvez você também queira consultar **FirstSortedValue** e **rangemin**, que têm uma operação semelhante à função **Min**.*

Sintaxe:

```
Min({[SetExpression] [TOTAL [<fld {,fld}>]]} expr [,rank])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
expr	A expressão ou campo que contém os dados a serem medidos.
rank	O valor padrão de **rank** é 1, que corresponde ao valor mais baixo. Se **rank** for especificado com 2, o segundo valor mais baixo será retornado. Se **rank** for 3, o terceiro valor mais baixo será retornado, e assim por diante.
SetExpression	Por padrão, a função de agregação agregará um conjunto de registros possíveis definidos pela seleção. Um conjunto de registros alternativos pode ser definido por uma expressão de análise.
TOTAL	Se a palavra **TOTAL** ocorrer antes dos argumentos da função, o cálculo será feito sobre todos os valores possíveis, dadas as seleções atuais, e não apenas sobre os pertinentes ao valor dimensional atual, isto é, desconsiderando as dimensões do gráfico. Usando **TOTAL [<fld {.fld}>]**, em que o qualificador **TOTAL** é seguido por uma lista de um ou mais nomes de campo como um subconjunto das variáveis de dimensão de gráfico, você cria um subconjunto dos valores possíveis totais.

Exemplos e resultados:

Dados

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

A função Min() deve retornar um valor não NULL da matriz de valores especificados pela expressão, se houver. Portanto, nos exemplos, como há NULL valores nos dados, a função retorna o primeiro valor não NULL avaliado da expressão.

Exemplos e resultados

Exemplos	Resultados
`Min(UnitSales)`	2, porque esse é o valor não NULL mais baixo em `UnitSales`.
O valor de um pedido é calculado a partir do número de unidades vendidas em (`UnitSales`), multiplicado pelo preço unitário. `Min(UnitSales*UnitPrice)`	40, porque esse é o valor não NULL mais baixo do resultado do cálculo de todos os valores possíveis de (`UnitSales`)*(`UnitPrice`).
`Min(UnitSales, 2)`	4, que é o segundo valor mais baixo (depois dos valores NULL).
`Min(TOTAL UnitSales)`	2, porque o qualificador TOTAL significa que o menor valor possível é encontrado, desconsiderando as dimensões do gráfico. Para um gráfico com Customer como a dimensão, o qualificador TOTAL garantirá que o valor mínimo em todo o conjunto de dados seja retornado, em vez da UnitSales mínima para cada cliente.
Faça a seleção Customer B. `Min({1} TOTAL UnitSales)`	2, que é independente da seleção de Customer B. A expressão Set Analysis {1} define o conjunto de registros a serem avaliados como ALL, independentemente da seleção feita.

Dados usados nos exemplos:

```
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 '|');
```

Consulte também:

- *FirstSortedValue - função de gráfico (page 334)*
- *RangeMin (page 1355)*

Mode

Mode() retorna o valor que ocorre com mais frequência, o valor modal, dos dados agregados na expressão, conforme definido por uma cláusula **group by**. A função **Mode()** pode retornar valores numéricos além de valores de texto.

Sintaxe:

```
Mode ( expr)
```

Tipo de dados de retorno: dual

Argumentos

Argumento	Descrição
expr Expression	A expressão ou campo que contém os dados a serem medidos.

Limitações:

Se mais de um valor ocorrer com a mesma frequência, retornará NULL.

Exemplos e resultados:

Adicione o script de exemplo ao seu aplicativo e execute-o. Em seguida, adicione pelo menos os campos listados na coluna de resultados a uma pasta para ver o resultado.

Para obter a mesma aparência que na coluna de resultados abaixo, no painel de propriedades, em Classificação, alterne de Automática para Personalizada e desmarque a classificação numérica e alfabética.

Exemplos de scripts

Exemplo	Resultado
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 porque AA é o único produto vendido mais de uma vez.

Mode - função de gráfico

Mode() encontra o valor mais geralmente ocorrido, o valor do modo, nos dados agregados. A função **Mode()** pode processar valores de texto, além de valores numéricos.

Sintaxe:

```
Mode({[SetExpression] [TOTAL [<fld {,fld}>]]} expr)
```

Tipo de dados de retorno: dual

Argumentos:

Argumentos

Argumento	Descrição
expr	A expressão ou campo que contém os dados a serem medidos.
SetExpression	Por padrão, a função de agregação agregará um conjunto de registros possíveis definidos pela seleção. Um conjunto de registros alternativos pode ser definido por uma expressão de análise.
TOTAL	Se a palavra **TOTAL** ocorrer antes dos argumentos da função, o cálculo será feito sobre todos os valores possíveis, dadas as seleções atuais, e não apenas sobre os pertinentes ao valor dimensional atual, isto é, desconsiderando as dimensões do gráfico. Usando **TOTAL [<fld {.fld}>]**, em que o qualificador **TOTAL** é seguido por uma lista de um ou mais nomes de campo como um subconjunto das variáveis de dimensão de gráfico, você cria um subconjunto dos valores possíveis totais.

Exemplos e resultados:

Dados

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

Exemplos e resultados

Exemplos	Resultados
`Mode(UnitPrice)` Faça a seleção `Customer A`.	15, porque esse é o valor que ocorre mais comumente em `UnitSales`. Retorna NULL (-). Nenhum valor único ocorre mais frequentemente que o outro.
`Mode(Product)` Faça a seleção `Customer A`.	AA, porque esse é o valor que ocorre mais comumente em `Product`. Retorna NULL (-). Nenhum valor único ocorre mais frequentemente que o outro.
`Mode (TOTAL UnitPrice)`	15, porque o qualificador TOTAL significa que o valor que ocorre mais comumente ainda é 15, desconsiderando as dimensões do gráfico.
Faça a seleção `Customer B`. `Mode({1} TOTAL UnitPrice)`	15, independentemente da seleção feita, pois a expressão Set Analysis {1} define o conjunto de registros a ser avaliado como ALL, independente da seleção feita.

Dados usados nos exemplos:

```
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 '|');
```

Consulte também:

- *Avg - função de gráfico (page 404)*
- *Median - função de gráfico (page 443)*

Only

Only() retorna um valor se houver um e somente um resultado possível a partir dos dados agregados. Se os registros contiverem apenas um valor, esse valor é retornado; caso contrário, NULL é retornado. Use a cláusula **group by** para avaliar múltiplos registros. A função **Only()** pode retornar valores numéricos e de texto.

Sintaxe:

```
Only ( expr )
```

Tipo de dados de retorno: dual

Argumentos

Argumento	Descrição
expr Expression	A expressão ou campo que contém os dados a serem medidos.

Exemplos e resultados:

Adicione o script de exemplo ao seu aplicativo e execute-o. Em seguida, adicione pelo menos os campos listados na coluna de resultados a uma pasta para ver o resultado.

Para obter a mesma aparência que na coluna de resultados abaixo, no painel de propriedades, em Classificação, alterne de Automática para Personalizada e desmarque a classificação numérica e alfabética.

```
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;
```

Tabela resultante

Customer	MyUniqIDCheck
Astrida	1 porque apenas o cliente Astrida tem registros completos que incluem CustomerID.

Only - função de gráfico

Only() retorna um valor se houver um e somente um resultado possível a partir dos dados agregados. Por exemplo, procurar o único produto em que o preço unitário =9 retornará NULL se mais de um produto tiver um preço unitário de 9.

Sintaxe:

```
Only([{SetExpression}] [TOTAL [<fld {,fld}>]] expr)
```

Tipo de dados de retorno: dual

Argumentos:

Argumentos

Argumento	Descrição
expr	A expressão ou campo que contém os dados a serem medidos.
SetExpression	Por padrão, a função de agregação agregará um conjunto de registros possíveis definidos pela seleção. Um conjunto de registros alternativos pode ser definido por uma expressão de análise.
TOTAL	Se a palavra **TOTAL** ocorrer antes dos argumentos da função, o cálculo será feito sobre todos os valores possíveis, dadas as seleções atuais, e não apenas sobre os pertinentes ao valor dimensional atual, isto é, desconsiderando as dimensões do gráfico. Usando **TOTAL [<fld {.fld}>]**, em que o qualificador **TOTAL** é seguido por uma lista de um ou mais nomes de campo como um subconjunto das variáveis de dimensão de gráfico, você cria um subconjunto dos valores possíveis totais.

Use Only() quando desejar obter um resultado NULL, se houver vários valores possíveis nos dados de exemplo.

Exemplos e resultados:

Dados

Customer	Product	UnitSales	UnitPrice
Astrida	AA	4	16

Customer	Product	UnitSales	UnitPrice
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

Exemplos e resultados

Exemplos	Resultados
`Only({<UnitPrice={9}>} Product)`	BB, porque esse é o único `Product`que tem um `UnitPrice` de '9'.
`Only({<Product={DD}>} Customer)`	Betacab, porque é o único `Customer` vendendo um `Product` chamado 'DD'.
`Only({<UnitPrice={20}>} UnitSales)`	O número de `UnitSales` onde `UnitPrice` é 20 é 2, pois existe apenas um único valor de `UnitSales`, onde o `UnitPrice` =20.
`Only({<UnitPrice={15}>} UnitSales)`	NULL, pois existem dois valores de `UnitSales`, onde o `UnitPrice` =15.

Dados usados nos exemplos:

```
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() calcula o total dos valores agregados na expressão, conforme definido pela cláusula **group by**.

Sintaxe:

```
sum ( [ distinct] expr)
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
distinct	Se a palavra **distinct** aparecer antes da expressão, todas as duplicatas serão ignoradas.
expr Expression	A expressão ou campo que contém os dados a serem medidos.

Exemplos e resultados:

Adicione o script de exemplo ao seu aplicativo e execute-o. Em seguida, adicione pelo menos os campos listados na coluna de resultados a uma pasta para ver o resultado.

Para obter a mesma aparência que na coluna de resultados abaixo, no painel de propriedades, em Classificação, alterne de Automática para Personalizada e desmarque a classificação numérica e alfabética.

```
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;
```

Tabela resultante

Customer	MySum
Astrida	39
Betacab	9
Canutility	8

Sum - função de gráfico

Sum() calcula o total dos valores dados pela expressão ou campo em todos os dados agregados.

Sintaxe:

```
Sum([{SetExpression}] [DISTINCT] [TOTAL [<fld {,fld}>]] expr)
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
expr	A expressão ou campo que contém os dados a serem medidos.
SetExpression	Por padrão, a função de agregação agregará um conjunto de registros possíveis definidos pela seleção. Um conjunto de registros alternativos pode ser definido por uma expressão de análise.
DISTINCT	Se a palavra **DISTINCT** aparecer antes dos argumentos de função, as duplicatas resultantes da avaliação dos argumentos de função serão ignoradas. *Embora o qualificador DISTINCT não seja suportado, use-o com extrema cautela, já que ele poderá confundir o leitor a pensar que o valor total é mostrado quando alguns dados foram omitidos.*
TOTAL	Se a palavra **TOTAL** ocorrer antes dos argumentos da função, o cálculo será feito sobre todos os valores possíveis, dadas as seleções atuais, e não apenas sobre os pertinentes ao valor dimensional atual, isto é, desconsiderando as dimensões do gráfico. Usando **TOTAL [<fld {.fld}>]**, em que o qualificador **TOTAL** é seguido por uma lista de um ou mais nomes de campo como um subconjunto das variáveis de dimensão de gráfico, você cria um subconjunto dos valores possíveis totais.

Exemplos e resultados:

Dados

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

Exemplos e resultados

Exemplos	Resultados
`Sum(UnitSales)`	38. O total dos valores em `UnitSales`.
`Sum(UnitSales*UnitPrice)`	505. O total de `UnitPrice` multiplicado pelos `UnitSales` agregados.
`Sum (TOTAL UnitSales*UnitPrice)`	505 para todas as linhas da tabela, além do total, pois o qualificador TOTAL significa que a soma ainda é 505, desconsiderando as dimensões do gráfico.
Faça a seleção `Customer B`. `Sum({1} TOTAL UnitSales*UnitPrice)`	505, independentemente da seleção feita, pois a expressão Set Analysis {1} define o conjunto de registros a ser avaliado como ALL, independente da seleção feita.

Dados usados nos exemplos:

```
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 '|');
```

Funções de agregação de contador

As funções de agregação de contador retornam vários tipos de contagens de uma expressão para um número de registros em um script de carregamento de dados ou um número de valores em uma dimensão de gráfico.

Cada função é descrita adicionalmente após a visão geral. Você também pode clicar no nome da função na sintaxe para acessar imediatamente os detalhes dessa função específica.

Funções de agregação de contador no script de carregamento de dados

Count

Count() retorna o número valores agregados na expressão, conforme definido por uma cláusula **group by**.

```
Count ([distinct ] expression | * )
```

MissingCount

MissingCount() retorna o número de valores agregados que faltam na expressão, como definido por uma cláusula **group by**.

```
MissingCount ([ distinct ] expression)
```

NullCount
NullCount() retorna o número de valores NULL agregados na expressão, como definido por uma cláusula **group by**.

```
NullCount ([ distinct ] expression)
```

NumericCount
NumericCount() returna o número de valores numéricos encontrados na expressão, como definido por uma cláusula **group by**.

```
NumericCount ([ distinct ] expression)
```

TextCount
TextCount() retorna o número de valores de campo não numéricos agregados na expressão, como definido por uma cláusula **group by**.

```
TextCount ([ distinct ] expression)
```

Funções de agregação de contador em expressões do gráfico

As seguintes funções de agregação de contador podem ser usadas nos seguintes mapas.

Count
Count() é utilizada para agregar o número de valores, de texto e numérico em cada uma das dimensões de gráfico.

```
Count - função de gráfico({[SetExpression] [DISTINCT] [TOTAL [<fld {,fld}>]]}
expr)
```

MissingCount
MissingCount() é utilizada para agregar o número de valores ausentes em cada dimensão de gráfico. Todos os valores ausentes são valores não numéricos.

```
MissingCount - função de gráfico({[SetExpression] [DISTINCT] [TOTAL [<fld
{,fld}>]] expr)
```

NullCount
NullCount() é utilizada para agregar o número de valores NULL em cada dimensão de gráfico.

```
NullCount - função de gráfico({[SetExpression][DISTINCT] [TOTAL [<fld
{,fld}>]]} expr)
```

NumericCount
NumericCount() agrega o número de valores numéricos em cada dimensão de gráfico.

```
NumericCount - função de gráfico({[SetExpression] [DISTINCT] [TOTAL [<fld
{,fld}>]]} expr)
```

TextCount
TextCount() é utilizada para agregar o número de valores de campo que são não numéricos em cada dimensão de gráfico.

```
TextCount - função de gráfico({[SetExpression] [DISTINCT] [TOTAL [<fld
{,fld}>]]} expr)
```

Count

Count() retorna o número valores agregados na expressão, conforme definido por uma cláusula **group by**.

Sintaxe:

```
Count( [distinct ] expr)
```

Tipo de dados de retorno: inteiro

Argumentos:

Argumentos

Argumento	Descrição
expr	A expressão ou campo que contém os dados a serem medidos.
distinct	Se a palavra **distinct** aparecer antes da expressão, todas as duplicatas serão ignoradas.

Exemplos e resultados:

Adicione o script de exemplo ao seu aplicativo e execute-o. Em seguida, adicione pelo menos os campos listados na coluna de resultados a uma pasta para ver o resultado.

Para obter a mesma aparência que na coluna de resultados abaixo, no painel de propriedades, em Classificação, alterne de Automática para Personalizada e desmarque a classificação numérica e alfabética.

Exemplos de scripts

Exemplo	Resultado
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 Contanto que a dimensão Customer seja incluída na tabela da pasta; caso contrário, o resultado de OrdersByCustomer será 3, 2.
Dado que a tabela **Temp** é carregada como no exemplo anterior: LOAD Count(OrderNumber) as TotalOrderNumber Resident Temp;	TotalOrderNumber 10
Dado que a tabela **Temp** é carregada como no primeiro exemplo: LOAD Count(distinct OrderNumber) as TotalOrderNumber Resident Temp;	TotalOrderNumber 8 Como há dois valores de OrderNumber com o mesmo valor, 1, e um valor nulo.

Count - função de gráfico

Count() é utilizada para agregar o número de valores, de texto e numérico em cada uma das dimensões de gráfico.

Sintaxe:

```
Count({[SetExpression] [DISTINCT] [TOTAL [<fld {,fld}>]]} expr)
```

Tipo de dados de retorno: inteiro

Argumentos:

Argumentos

Argumento	Descrição
expr	A expressão ou campo que contém os dados a serem medidos.

Argumento	Descrição
SetExpression	Por padrão, a função de agregação agregará um conjunto de registros possíveis definidos pela seleção. Um conjunto de registros alternativos pode ser definido por uma expressão de análise.
DISTINCT	Se a palavra **DISTINCT** aparecer antes dos argumentos de função, as duplicatas resultantes da avaliação dos argumentos de função serão ignoradas.
TOTAL	Se a palavra **TOTAL** ocorrer antes dos argumentos da função, o cálculo será feito sobre todos os valores possíveis, dadas as seleções atuais, e não apenas sobre os pertinentes ao valor dimensional atual, isto é, desconsiderando as dimensões do gráfico. Usando **TOTAL [<fld {.fld}>]**, em que o qualificador **TOTAL** é seguido por uma lista de um ou mais nomes de campo como um subconjunto das variáveis de dimensão de gráfico, você cria um subconjunto dos valores possíveis totais.

Exemplos e resultados:

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

Os exemplos a seguir assumem que todos os clientes são selecionados, exceto onde indicado.

Exemplos e resultados

Exemplo	Resultado
`Count(OrderNumber)`	10, porque existem 10 campos que podem ter um valor para OrderNumber, e todos os registros, até mesmo os vazios, são contados. *"0" conta como um valor, e não uma célula vazia. No entanto, se uma lista agregar ao 0 para uma dimensão, esta não será incluída nas tabelas.*
`Count(Customer)`	10, porque Count avalia o número de ocorrências em todos os campos.
`Count(DISTINCT [Customer])`	4, porque usar o qualificador Distinct, Count apenas avalia as ocorrências únicas.
Com a condição de que o cliente Canutility seja selecionado `Count(OrderNumber)/Count ({1} TOTAL OrderNumber)`	0,2, porque a expressão retorna o número de ordens do cliente selecionado como uma porcentagem das ordens de todos os clientes. Nesse caso, 2/10.
Dado que os clientes astrida e Canutility sejam selecionados `Count(TOTAL <Product> OrderNumber)`	5, porque esse é o número de ordens colocadas em produtos apenas para os clientes selecionados e as células vazias são contadas.

Dados usados nos exemplos:

```
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() retorna o número de valores agregados que faltam na expressão, como definido por uma cláusula **group by**.

Sintaxe:

```
MissingCount ( [ distinct ] expr)
```

Tipo de dados de retorno: inteiro

Argumentos:

Argumentos

Argumento	Descrição
expr Expression	A expressão ou campo que contém os dados a serem medidos.
distinct	Se a palavra **distinct** aparecer antes da expressão, todas as duplicatas serão ignoradas.

Exemplos e resultados:

Adicione o script de exemplo ao seu aplicativo e execute-o. Em seguida, adicione pelo menos os campos listados na coluna de resultados a uma pasta para ver o resultado.

Para obter a mesma aparência que na coluna de resultados abaixo, no painel de propriedades, em Classificação, alterne de Automática para Personalizada e desmarque a classificação numérica e alfabética.

Exemplos de scripts

Exemplo	Resultado
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 O segundo comando fornece: TotalMissingCount 3 em uma tabela com essa dimensão.
Dado que a tabela **Temp** é carregada como no exemplo anterior: LOAD MissingCount(distinct OrderNumber) as TotalMissingCountDistinct Resident Temp;	TotalMissingCountDistinct 1 Porque só há um OrderNumber valor ausente.

MissingCount - função de gráfico

MissingCount() é utilizada para agregar o número de valores ausentes em cada dimensão de gráfico. Todos os valores ausentes são valores não numéricos.

Sintaxe:

```
MissingCount({[SetExpression] [DISTINCT] [TOTAL [<fld {,fld}>]]} expr)
```

Tipo de dados de retorno: inteiro

Argumentos:

Argumentos

Argumento	Descrição
expr	A expressão ou campo que contém os dados a serem medidos.
SetExpression	Por padrão, a função de agregação agregará um conjunto de registros possíveis definidos pela seleção. Um conjunto de registros alternativos pode ser definido por uma expressão de análise.
DISTINCT	Se a palavra **DISTINCT** aparecer antes dos argumentos de função, as duplicatas resultantes da avaliação dos argumentos de função serão ignoradas.
TOTAL	Se a palavra **TOTAL** ocorrer antes dos argumentos da função, o cálculo será feito sobre todos os valores possíveis, dadas as seleções atuais, e não apenas sobre os pertinentes ao valor dimensional atual, isto é, desconsiderando as dimensões do gráfico. Usando **TOTAL [<fld {.fld}>]**, em que o qualificador **TOTAL** é seguido por uma lista de um ou mais nomes de campo como um subconjunto das variáveis de dimensão de gráfico, você cria um subconjunto dos valores possíveis totais.

Exemplos e resultados:

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

Customer	Product	OrderNumber	UnitSales	Unit Price
Canutility	CC			19
Divadip	AA	2	4	16
Divadip	DD	3		25

Exemplos e resultados

Exemplo	Resultado
`MissingCount([OrderNumber])`	3, pois 3 dos 10 campos OrderNumber estão vazios *"0" conta como um valor, e não uma célula vazia. No entanto, se uma lista agregar ao 0 para uma dimensão, esta não será incluída nas tabelas.*
`MissingCount ([OrderNumber])/MissingCount ({1} Total [OrderNumber])`	A expressão retorna o número de pedidos incompletos do cliente selecionado como uma fração dos pedidos incompletos de todos os clientes. Existe um total de 3 valores ausentes para OrderNumber para todos os clientes. Então, para cada Customer que tem um valor ausente para Product o resultado é 1/3.

Dados usados no exemplo:

```
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() retorna o número de valores NULL agregados na expressão, como definido por uma cláusula **group by**.

Sintaxe:

```
NullCount ( [ distinct ] expr)
```

Tipo de dados de retorno: inteiro

Argumentos:

Argumentos

Argumento	Descrição
expr Expression	A expressão ou campo que contém os dados a serem medidos.
distinct	Se a palavra **distinct** aparecer antes da expressão, todas as duplicatas serão ignoradas.

Exemplos e resultados:

Adicione o script de exemplo ao seu aplicativo e execute-o. Em seguida, adicione pelo menos os campos listados na coluna de resultados a uma pasta para ver o resultado.

Para obter a mesma aparência que na coluna de resultados abaixo, no painel de propriedades, em Classificação, alterne de Automática para Personalizada e desmarque a classificação numérica e alfabética.

Exemplos de scripts

Exemplo	Resultado

`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`
	
	O segundo comando fornece:
	`TotalNullCount`
	
	`1`
	em uma tabela com essa dimensão, pois apenas um registro contém um valor nulo.

NullCount - função de gráfico

NullCount() é utilizada para agregar o número de valores NULL em cada dimensão de gráfico.

Sintaxe:

```
NullCount({[SetExpression][DISTINCT] [TOTAL [<fld {,fld}>]]} expr)
```

Tipo de dados de retorno: inteiro

Argumentos:

Argumentos

Argumento	Descrição
expr	A expressão ou campo que contém os dados a serem medidos.
set_ expression	Por padrão, a função de agregação agregará um conjunto de registros possíveis definidos pela seleção. Um conjunto de registros alternativos pode ser definido por uma expressão de análise.
DISTINCT	Se a palavra **DISTINCT** aparecer antes dos argumentos de função, as duplicatas resultantes da avaliação dos argumentos de função serão ignoradas.
TOTAL	Se a palavra **TOTAL** ocorrer antes dos argumentos da função, o cálculo será feito sobre todos os valores possíveis, dadas as seleções atuais, e não apenas sobre os pertinentes ao valor dimensional atual, isto é, desconsiderando as dimensões do gráfico. Usando **TOTAL [<fld {.fld}>]**, em que o qualificador **TOTAL** é seguido por uma lista de um ou mais nomes de campo como um subconjunto das variáveis de dimensão de gráfico, você cria um subconjunto dos valores possíveis totais.

Exemplos e resultados:

Exemplos e resultados

Exemplo	Resultado
NullCount ([OrderNumber])	1, pois introduzimos um valor nulo usando NullInterpret no comando inline **LOAD**.

Dados usados no exemplo:

```
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() returna o número de valores numéricos encontrados na expressão, como definido por uma cláusula **group by**.

Sintaxe:

```
NumericCount ( [ distinct ] expr)
```

Tipo de dados de retorno: inteiro

Argumentos:

Argumentos

Argumento	Descrição
expr Expression	A expressão ou campo que contém os dados a serem medidos.
distinct	Se a palavra **distinct** aparecer antes da expressão, todas as duplicatas serão ignoradas.

Exemplos e resultados:

Adicione o script de exemplo ao seu aplicativo e execute-o. Em seguida, adicione pelo menos os campos listados na coluna de resultados a uma pasta para ver o resultado.

Para obter a mesma aparência que na coluna de resultados abaixo, no painel de propriedades, em Classificação, alterne de Automática para Personalizada e desmarque a classificação numérica e alfabética.

Exemplo de scripts

Exemplo	Resultado
`LOAD NumericCount(OrderNumber) as TotalNumericCount Resident Temp;`	O segundo comando fornece: TotalNumericCount 7 em uma tabela com essa dimensão.
Dado que a tabela **Temp** é carregada como no exemplo anterior: `LOAD NumericCount(distinct OrderNumber) as TotalNumericCountDistinct Resident Temp;`	TotalNumericCountDistinct 6 Como há um OrderNumber que duplica o outro, então o resultado é 6, que não são duplicatas.

Exemplo:

```
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;
```

Tabela resultante

Customer	NumericCountByCustomer
Astrida	3
Betacab	2
Canutility	0
Divadip	2

NumericCount - função de gráfico

NumericCount() agrega o número de valores numéricos em cada dimensão de gráfico.

Sintaxe:

```
NumericCount({[SetExpression] [DISTINCT] [TOTAL [<fld {,fld}>]]} expr)
```

Tipo de dados de retorno: inteiro

Argumentos:

Argumentos

Argumento	Descrição
expr	A expressão ou campo que contém os dados a serem medidos.

Argumento	Descrição
set_ expression	Por padrão, a função de agregação agregará um conjunto de registros possíveis definidos pela seleção. Um conjunto de registros alternativos pode ser definido por uma expressão de análise.
DISTINCT	Se a palavra **DISTINCT** aparecer antes dos argumentos de função, as duplicatas resultantes da avaliação dos argumentos de função serão ignoradas.
TOTAL	Se a palavra **TOTAL** ocorrer antes dos argumentos da função, o cálculo será feito sobre todos os valores possíveis, dadas as seleções atuais, e não apenas sobre os pertinentes ao valor dimensional atual, isto é, desconsiderando as dimensões do gráfico. Usando **TOTAL [<fld {.fld}>]**, em que o qualificador **TOTAL** é seguido por uma lista de um ou mais nomes de campo como um subconjunto das variáveis de dimensão de gráfico, você cria um subconjunto dos valores possíveis totais.

Exemplos e resultados:

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

Os exemplos a seguir assumem que todos os clientes são selecionados, exceto onde indicado.

Exemplos e resultados

Exemplo	Resultado
`NumericCount ([OrderNumber])`	7 porque três dos 10 campos em OrderNumber estão vazios. *"0" conta como um valor, e não uma célula vazia. No entanto, se uma lista agregar ao 0 para uma dimensão, esta não será incluída nas tabelas.*
`NumericCount ([Product])`	0 porque todos os nomes de produtos são de texto. Normalmente, você pode usar isso para verificar se nenhum campo de texto recebeu conteúdo numérico.
`NumericCount (DISTINCT [OrderNumber])/Count (DISTINCT [OrderNumber)]`	Conta todo o número dos números de ordem numérica distintos e divide pelo número de números de ordem numéricos e não numéricos. Será 1 se todos os valores de campo forem numéricos. Normalmente, você pode usar isso para verificar se todos os valores de campo são numéricos. No exemplo, existem 7 valores numéricos distintos para OrderNumber de 8 numéricos e não numéricos distintos, então a expressão retorna 0,875.

Dados usados no exemplo:

```
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() retorna o número de valores de campo não numéricos agregados na expressão, como definido por uma cláusula **group by**.

Sintaxe:

```
TextCount ( [ distinct ] expr)
```

Tipo de dados de retorno: inteiro

Argumentos:

Argumentos

Argumento	Descrição
expr Expression	A expressão ou campo que contém os dados a serem medidos.
distinct	Se a palavra **distinct** aparecer antes da expressão, todas as duplicatas serão ignoradas.

Exemplos e resultados:

Adicione o script de exemplo ao seu aplicativo e execute-o. Em seguida, adicione pelo menos os campos listados na coluna de resultados a uma pasta para ver o resultado.

Para obter a mesma aparência que na coluna de resultados abaixo, no painel de propriedades, em Classificação, alterne de Automática para Personalizada e desmarque a classificação numérica e alfabética.

Exemplo:

```
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;
```

Tabela resultante

Customer	ProductTextCount
Astrida	3
Betacab	3
Canutility	2
Divadip	2

Exemplo:

```
LOAD Customer,TextCount(OrderNumber) as OrderNumberTextCount Resident Temp Group By Customer;
```

Tabela resultante

Customer	OrderNumberTextCount
Astrida	0
Betacab	1
Canutility	2
Divadip	0

TextCount - função de gráfico

TextCount() é utilizada para agregar o número de valores de campo que são não numéricos em cada dimensão de gráfico.

Sintaxe:

```
TextCount({[SetExpression] [DISTINCT] [TOTAL [<fld {,fld}>]]} expr)
```

Tipo de dados de retorno: inteiro

Argumentos:

Argumentos

Argumento	Descrição
expr	A expressão ou campo que contém os dados a serem medidos.
SetExpression	Por padrão, a função de agregação agregará um conjunto de registros possíveis definidos pela seleção. Um conjunto de registros alternativos pode ser definido por uma expressão de análise.
DISTINCT	Se a palavra **DISTINCT** aparecer antes dos argumentos de função, as duplicatas resultantes da avaliação dos argumentos de função serão ignoradas.
TOTAL	Se a palavra **TOTAL** ocorrer antes dos argumentos da função, o cálculo será feito sobre todos os valores possíveis, dadas as seleções atuais, e não apenas sobre os pertinentes ao valor dimensional atual, isto é, desconsiderando as dimensões do gráfico. Usando **TOTAL [<fld {.fld}>]**, em que o qualificador **TOTAL** é seguido por uma lista de um ou mais nomes de campo como um subconjunto das variáveis de dimensão de gráfico, você cria um subconjunto dos valores possíveis totais.

Exemplos e resultados:

Data

Customer	Product	OrderNumber	UnitSales	Unit Price
Astrida	AA	1	4	16

Customer	Product	OrderNumber	UnitSales	Unit Price
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

Exemplos e resultados

Exemplo	Resultado
`TextCount ([Product])`	10 porque todos os 10 campos em Product são de texto. *"0" conta como um valor, e não uma célula vazia. No entanto, se uma lista agregar ao 0 para uma dimensão, esta não será incluída nas tabelas. Células vazias são avaliadas como sendo não texto e não são contadas pelo TextCount.*
`TextCount ([OrderNumber])`	3 porque as células vazias são contadas. Normalmente, você usa isso para verificar se nenhum campo numérico recebeu valores de texto ou se são não zero.
`TextCount (DISTINCT [Product])/Count ([Product)]`	Conta todo o número de valores de texto distintos de Product (4), e divide pelo número total de valores em Product (10). O resultado é 0,4.

Dados usados no exemplo:

```
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 '|');
```

Funções de agregação financeiras

Esta seção descreve as funções de agregação das operações financeiras referentes a pagamentos e fluxo de caixa.

Cada função é descrita adicionalmente após a visão geral. Você também pode clicar no nome da função na sintaxe para acessar imediatamente os detalhes dessa função específica.

Funções de agregação financeira no script de carregamento de dados

IRR

IRR() retorna a taxa interna agregada de retorno para uma série de fluxos de caixa representados por números na expressão com iterações em vários registros, conforme definido por uma cláusula group by.

```
IRR (expression)
```

XIRR

XIRR() retorna a taxa interna de retorno agregada (anual) para um cronograma de fluxos de caixa (que não é necessariamente periódico) representado por números emparelhados em **pmt** e **date** iterados sobre um número de registros, conforme definido por uma cláusula group by. Todos os pagamentos têm descontos baseados em um ano de 365 dias.

```
XIRR (valueexpression, dateexpression )
```

NPV

A função de script **NPV()** tem uma taxa de desconto e vários valores ordenados por período. As entradas (receitas) são positivas e as saídas (pagamentos futuros) são consideradas valores negativos para esses cálculos. Elas ocorrem no final de cada período.

```
NPV (rate, expression)
```

XNPV

XNPV() retorna o valor presente líquido agregado para um cronograma de fluxos de caixa (não necessariamente periódicos) representados por números emparelhados em **pmt** e **date**. Todos os pagamentos têm descontos baseados em um ano de 365 dias.

```
XNPV (rate, valueexpression, dateexpression)
```

Funções de agregação financeira em expressões do gráfico

Estas funções de agregação financeiras podem ser usadas em gráficos.

IRR

IRR() retorna a taxa interna agregada de uma série de fluxos de caixa representados pelos números na expressão dada pelo **value** repetido nas dimensões de gráfico.

```
IRR - função de gráfico[TOTAL [<fld {,fld}>]] value)
```

NPV

NPV() retorna o valor líquido agregado de um investimento com base em uma **discount_rate** por período e em uma série de pagamentos futuros (valores negativos) e receitas (valores positivos) representados pelos números no **value** com iterações nas dimensões do gráfico. Assume-se que os pagamentos e as receitas ocorram no final de cada período.

```
NPV - função de gráfico([TOTAL [<fld {,fld}>]] discount_rate, value)
```

XIRR

XIRR() retorna a taxa interna de retorno agregada (anual) para um cronograma de fluxos de caixa (que não é necessariamente periódico) representado por números emparelhados nas expressões especificadas por **pmt** e **date** iterados sobre as dimensões do gráfico. Todos os pagamentos têm descontos baseados em um ano de 365 dias.

```
XIRR - função de gráfico([TOTAL [<fld {,fld}>]] pmt, date)
```

XNPV

XNPV() retorna o valor líquido atual agregado de uma programação de fluxos de caixa (não necessariamente periódica) representados por números emparelhados nas expressões dadas por **pmt** e **date** repetidos nas dimensões de gráfico. Todos os pagamentos têm descontos baseados em um ano de 365 dias.

```
XNPV - função de gráfico([TOTAL [<fld{,fld}>]] discount_rate, pmt, date)
```

IRR

IRR() retorna a taxa interna agregada de retorno para uma série de fluxos de caixa representados por números na expressão com iterações em vários registros, conforme definido por uma cláusula group by.

Esses fluxos de caixa não precisam ser nivelados como seriam para uma anuidade. No entanto, os fluxos de caixa devem ocorrer em intervalos regulares, mensalmente ou anualmente por exemplo. A taxa de retorno interno é a taxa de juros recebida em um investimento que consiste em pagamentos (valores negativos) e receita (valores positivos) que ocorrem em períodos regulares. A função precisa de um valor positivo e um valor negativo, pelo menos, para ser calculada.

Essa função usa uma versão simplificada do método de Newton para calcular a taxa de retorno interna (TIR).

Sintaxe:

```
IRR(value)
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
value	A expressão ou campo que contém os dados a serem medidos.

Limitações:

Os valores de texto, os valores NULL e os valores ausentes são ignorados.

Exemplos e resultados:

Adicione o script de exemplo ao seu aplicativo e execute-o. Para ver o resultado, adicione os campos listados na coluna de resultados a uma pasta no seu aplicativo.

Exemplos e resultados:

Exemplos e resultados

Exemplo	Ano	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 - função de gráfico

IRR() retorna a taxa interna agregada de uma série de fluxos de caixa representados pelos números na expressão dada pelo **value** repetido nas dimensões de gráfico.

Esses fluxos de caixa não precisam ser nivelados como seriam para uma anuidade. No entanto, os fluxos de caixa devem ocorrer em intervalos regulares, mensalmente ou anualmente por exemplo. A taxa de retorno interno é a taxa de juros recebida em um investimento que consiste em pagamentos (valores negativos) e receita (valores positivos) que ocorrem em períodos regulares. A função precisa de um valor positivo e um valor negativo, pelo menos, para ser calculada.

Essa função usa uma versão simplificada do método de Newton para calcular a taxa de retorno interna (TIR).

Sintaxe:

```
IRR([TOTAL [<fld {,fld}>]] value)
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
value	A expressão ou campo que contém os dados a serem medidos.
TOTAL	Se a palavra **TOTAL** ocorrer antes dos argumentos da função, o cálculo será feito sobre todos os valores possíveis, dadas as seleções atuais, e não apenas sobre os pertinentes ao valor dimensional atual, isto é, desconsiderando as dimensões do gráfico. Usando **TOTAL [<fld {.fld}>]**, em que o qualificador **TOTAL** é seguido por uma lista de um ou mais nomes de campo como um subconjunto das variáveis de dimensão de gráfico, você cria um subconjunto dos valores possíveis totais.

Limitações:

O parâmetro da função de agregação não deve conter outras funções de agregação, a menos que essas agregações internas contenham o qualificador **TOTAL**. Para agregações aninhadas mais avançadas, use a função avançada **Aggr** junto com uma dimensão especificada.

Os valores de texto, os valores NULL e os valores ausentes são ignorados.

Exemplos e resultados:

Exemplos e resultados

Exemplo	Resultado
`IRR (Payments)`	0.1634 Os pagamentos são considerados de natureza periódica, por exemplo mensal. *O campo Date é usado no exemplo XIRR onde os pagamentos podem ser não periódicos, desde que você forneça as datas em que os pagamentos foram feitos.*

Dados usados nos exemplos:

```
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 '|');
```

Consulte também:

- *XIRR - função de gráfico (page 384)*
- *Aggr - função de gráfico (page 552)*

NPV

A função de script **NPV()** tem uma taxa de desconto e vários valores ordenados por período. As entradas (receitas) são positivas e as saídas (pagamentos futuros) são consideradas valores negativos para esses cálculos. Elas ocorrem no final de cada período.

O Valor presente líquido, ou NPV, é usado para calcular o valor total atual de um fluxo futuro de fluxos de caixa. Para calcular o NPV, precisamos estimar os fluxos de caixa futuros para cada período e determinar a taxa de desconto correta. A função de script **NPV()** tem uma taxa de desconto e vários valores ordenados por período. As entradas (receitas) são positivas e as saídas (pagamentos futuros) são consideradas valores negativos para esses cálculos. Elas ocorrem no final de cada período.

Sintaxe:

```
NPV(discount_rate, value)
```

Tipo de dados de retorno: numérico. Por padrão, o resultado será formatado como moeda.

A fórmula para calcular o valor atual líquido é:

$$NPV = \sum_{t=1}^{n} \frac{R_t}{(1+i)^t}$$

na qual:

- R_t = Entradas e saídas líquidas de caixa durante um único período `t`
- `i` = Taxa de desconto ou retorno que poderia ser obtida em investimentos alternativos
- `t` = Número de períodos do cronômetro

Argumentos

Argumento	Descrição
discount_rate	**discount_rate** é a taxa percentual de desconto aplicado. Um valor de 0,1 indicaria uma taxa de desconto de 10%.
value	Esse campo contém valores para vários períodos ordenados por período. Supõe-se que o primeiro valor seja o fluxo de caixa no final do período 1, e assim por diante.

Limitações:

A função `NPV()` tem as seguintes limitações:

- Os valores de texto, os valores NULL e os valores ausentes são ignorados.
- Os valores do fluxo de caixa devem estar em ordem crescente de período.

Quando usar

`NPV()` é uma função financeira usada para verificar a rentabilidade do projeto e derivar outras medidas. Essa função é útil quando fluxos de caixa estão disponíveis como dados brutos.

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplo 1: Pagamento único (script)

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados de um projeto e seu fluxo de caixa referentes a um período, que é carregado em uma tabela denominada `CashFlow`.
- Um carregamento residente da tabela `CashFlow`, que é usado para calcular o campo NPV do projeto em uma tabela denominada `NPV`.
- Uma taxa de desconto codificada de 10%, usada no cálculo de NPV.
- Uma instrução `Group By`, que é usada para agrupar todos os pagamentos do projeto.

Script de carregamento

```
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;
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- PrjId
- NPV

Tabela de resultados

PrjId	NPV
1	$909.09

Para que um único pagamento de $ 1.000 seja recebido no final de um período, com uma taxa de desconto de 10% por período, o NPV é igual a $ 1.000 dividido por (1 + taxa de desconto). O NPV efetivo é igual a $ 909,09

Exemplo 2: Vários pagamentos (script)

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados de um projeto e seu fluxo de caixa referentes a vários períodos, que é carregado em uma tabela denominada CashFlow.
- Um carregamento residente da tabela CashFlow, que é usado para calcular o campo NPV do projeto em uma tabela denominada NPV.
- Uma taxa de desconto codificada de 10% (0,1) é usada no cálculo de NPV.
- Uma instrução Group By, que é usada para agrupar todos os pagamentos do projeto.

Script de carregamento

```
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;
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- PrjId
- NPV

Tabela de resultados

PrjId	NPV
1	$1735.54

Para pagamentos de $ 1.000 a serem recebidos no final de dois períodos, com uma taxa de desconto de 10% por período, o NPV efetivo é igual a $ 1.735,54.

Exemplo 3: Vários pagamentos (script)

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Taxas de desconto para dois projetos, que são carregadas em uma tabela denominada `Project`.
- Fluxos de caixa para vários períodos para cada projeto por ID de projeto e ID de período. Esse ID de período pode ser usado para ordenar os registros, caso os dados não estejam ordenados.
- A combinação de `NoConcatenate`, carregamentos residentes e a função `Left Join` para criar uma tabela temporária, `tmpNPV`. A tabela combina os registros das tabelas `Project` e `CashFlow` em uma tabela simples. Essa tabela terá taxas de desconto repetidas para cada período.
- Um carregamento residente da tabela `tmpNPV`, que é usado para calcular o campo NPV de cada projeto em uma tabela denominada `NPV`.
- A taxa de desconto de valor único associada a cada projeto. Isso é recuperado usando a função `only()` e é usado no cálculo de NPV para cada projeto.
- Uma instrução `Group By`, que é usada para agrupar todos os pagamentos de cada projeto por ID de projeto.

Para evitar que dados sintéticos ou redundantes sejam carregados no modelo de dados, a tabela `tmpNPV` é descartada no final do script.

Script de carregamento

```
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;
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- PrjId
- NPV

Tabela de resultados

PrjId	NPV
1	$2486.85
2	$2042.12

O ID de projeto 1 espera que pagamentos de $ 1.000 sejam recebidos ao final de três períodos, com uma taxa de desconto de 10% por período. Portanto, o NPV efetivo é igual a $ 2.486,85.

O ID de projeto 2 espera dois pagamentos de $ 500 e dois pagamentos adicionais de $ 1.000 em quatro períodos com uma taxa de desconto de 15%. Portanto, o NPV efetivo é igual a $ 2.042,12.

Exemplo 4: Exemplo de lucratividade do projeto (script)

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Taxas de desconto e investimentos iniciais (período 0) para dois projetos, carregados em uma tabela denominada `Project`.
- Fluxos de caixa para vários períodos para cada projeto por ID de projeto e ID de período. Esse ID de período pode ser usado para ordenar os registros, caso os dados não estejam ordenados.
- A combinação de `NoConcatenate`, carregamentos residentes e a função `Left Join` para criar uma tabela temporária, `tmpNPV`. A tabela combina os registros das tabelas `Project` e `CashFlow` em uma tabela simples. Essa tabela terá taxas de desconto repetidas para cada período.
- A taxa de desconto de valor único associada a cada projeto, que é recuperada usando a função `only()` e usada no cálculo do NPV para cada projeto.
- Um carregamento residente da tabela tmpNPV é usado para calcular o campo NPV de cada projeto em uma tabela denominada NPV.
- É criado um campo adicional que divide o NPV pelo investimento inicial de cada projeto para calcular o índice de lucratividade do projeto.
- Um agrupamento por extrato, agrupado por ID de projeto, é usado para agrupar todos os pagamentos de cada projeto.

Para evitar que dados sintéticos ou redundantes sejam carregados no modelo de dados, a tabela `tmpNPV` é descartada no final do script.

Script de carregamento

```
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;
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- PrjId
- NPV

Crie a seguinte medida:

```
=only(Profitability_Index)
```

Tabela de resultados

PrjId	NPV	=only(Profitability_Index)
1	$87039.82	0.87
2	$123513.71	1.24

O ID de projeto 1 tem um NPV efetivo de $ 87.039,82 e um investimento inicial de $ 100.000. Portanto, o índice de rentabilidade é igual a 0,87. Por ser menor que 1, o projeto não é lucrativo.

O ID de projeto 2 tem um NPV efetivo de $ 123.513,71 e um investimento inicial de $ 100.000. Portanto, o índice de rentabilidade é igual a 1,24. Por ser maior que 1, o projeto é lucrativo.

NPV - função de gráfico

NPV() retorna o valor líquido agregado de um investimento com base em uma **discount_rate** por período e em uma série de pagamentos futuros (valores negativos) e receitas (valores positivos) representados pelos números no **value** com iterações nas dimensões do gráfico. Assume-se que os pagamentos e as receitas ocorram no final de cada período.

Sintaxe:

```
NPV([TOTAL [<fld {,fld}>]] discount_rate, value)
```

Tipo de dados de retorno: numérico Por padrão, o resultado será formatado como moeda.

Argumentos:

Argumentos

Argumento	Descrição
discount_rate	**discount_rate** is the rate of discount over the length of the period.**discount_rate** é a taxa percentual de desconto aplicado.
value	A expressão ou campo que contém os dados a serem medidos.
TOTAL	Se a palavra **TOTAL** ocorrer antes dos argumentos da função, o cálculo será feito sobre todos os valores possíveis, dadas as seleções atuais, e não apenas sobre os pertinentes ao valor dimensional atual, isto é, desconsiderando as dimensões do gráfico. Usando **TOTAL [<fld {.fld}>]**, em que o qualificador **TOTAL** é seguido por uma lista de um ou mais nomes de campo como um subconjunto das variáveis de dimensão de gráfico, você cria um subconjunto dos valores possíveis totais. O qualificador **TOTAL** pode vir seguido de uma lista de um ou mais nomes de campos dentro de sinais de maior e menor que. Esses nomes de campos devem ser um subconjunto das variáveis de dimensões do gráfico. Nesse caso, o cálculo é feito ignorando-se todas as variáveis de dimensões do gráfico, exceto aquelas listadas, isto é, um valor será retornado para cada combinação de valores de campo nos campos de dimensão listados. Também podem ser incluídos na lista os campos que não forem uma dimensão em um gráfico no momento. Isso pode ser útil para grupos de dimensões em que os campos de dimensões não são fixos. A lista de todas as variáveis do grupo faz com que a função tenha efeito quando o nível hierárquico for alterado.

Limitações:

discount_rate e **value** não devem conter funções de agregação, a não ser que essas agregações internas contenham o qualificador **TOTAL**. Para agregações aninhadas mais avançadas, use a função avançada **Aggr** junto com uma dimensão especificada.

Os valores de texto, os valores NULL e os valores ausentes são ignorados.

Exemplos e resultados:

Exemplos e resultados

Exemplo	Resultado
`NPV(Discount, Payments)`	-$540.12

Dados usados nos exemplos:

```
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 '|');
```

Consulte também:

XNPV - função de gráfico (page 394)
Aggr - função de gráfico (page 552)

XIRR

XIRR() retorna a taxa interna de retorno agregada (anual) para um cronograma de fluxos de caixa (que não é necessariamente periódico) representado por números emparelhados em **pmt** e **date** iterados sobre um número de registros, conforme definido por uma cláusula group by. Todos os pagamentos têm descontos baseados em um ano de 365 dias.

Funcionalidade XIRR do Qlik (funções **XIRR()** e **RangeXIRR()**) usa a seguinte equação, resolvendo para o valor `Rate`, para determinar o valor XIRR correto:

```
XNPV(Rate, pmt, date) = 0
```

A equação é resolvida usando uma versão simplificada do método de Newton.

Sintaxe:

```
XIRR(pmt, date )
```

Tipo de dados de retorno: numérico

Argumentos

Argumento	Descrição
pmt	Pagamentos. A expressão ou campo que contém os fluxos de caixa correspondentes ao cronograma de pagamentos dado no **date**.
date	A expressão ou campo que contém o cronograma de datas correspondentes aos pagamentos de fluxo de caixa dados em **pmt**.

Ao trabalhar com essa função, as seguintes limitações são aplicáveis:

- Valores de texto, valores NULL e valores ausentes em qualquer uma das duas partes de um par de dados farão com que o par de dados inteiro seja ignorado.
- Essa função requer pelo menos um pagamento negativo válido e pelo menos um pagamento positivo válido (com datas válidas correspondentes). Se esses pagamentos não forem fornecidos, um valor NULL será retornado.

Estes tópicos podem ajudar você a trabalhar com essa função:

- *XNPV (page 388)*: Use essa função para calcular o valor presente líquido agregado para uma programação de fluxos de caixa.
- *RangeXIRR (page 1373)*: **RangeXIRR()** é a função de intervalo equivalente para a função **XIRR()**.

Em diferentes versões do Qlik Sense Client-Managed, há variações no algoritmo subjacente usado por essa função. Para obter mais informações sobre atualizações recentes do algoritmo, consulte o artigo de suporte Correção e atualização da função XIRR.

Exemplo

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Dados de transação para uma série de fluxos de caixa.
- O uso da função **XIRR()** para calcular a taxa de retorno anual interna para esses fluxos de caixa.

Script de carregamento

```
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;
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `Year`
- `XIRR2013`

Tabela de resultados

Ano	XIRR2013
2013	0.5385

Interpretando o valor de retorno XIRR

A funcionalidade XIRR geralmente é usada para analisar um investimento, em que há um pagamento de saída (negativo) no início e, posteriormente, uma série de pagamentos de receita menores (positivos). Veja a seguir um exemplo simplificado com apenas um pagamento negativo e um positivo:

```
Cashflow:
LOAD * inline [
Date|Payments
2023-01-01|-100
2024-01-01|110
] (delimiter is '|');
```

Fazemos um pagamento inicial de 100 e recebemos 110 de volta depois de exatamente um ano. Isso representa uma taxa de retorno de 10% ao ano. `XIRR(Payments, Date)` retorna um valor de 0,1.

O valor de retorno da funcionalidade XIRR pode ser positivo ou negativo. No caso de um investimento, um resultado negativo indica que o investimento é uma perda. A quantidade de ganho ou perda pode ser calculada com uma simples agregação de soma no campo de pagamentos.

No exemplo acima, estamos emprestando nosso dinheiro por um ano. A taxa de retorno pode ser considerada como juros. Também é possível usar a funcionalidade XIRR quando você está do outro lado da transação (por exemplo, se for o mutuário em vez do credor).

Considere este exemplo:

```
Cashflow:
LOAD * inline [
Date|Payments
2023-01-01|100
2024-01-01|-110
] (delimiter is '|');
```

Isso é igual ao primeiro exemplo, mas invertido. Aqui, tomamos 100 emprestados por um ano e pagamos com juros de 10%. Neste exemplo, o cálculo de XIRR retorna 0,1 (10%), o mesmo valor do primeiro exemplo.

Observe que, no primeiro exemplo, recebemos um lucro de 10 e, no segundo, tivemos uma perda de 10, mas o valor de retorno da funcionalidade XIRR é positivo para ambos os exemplos. Isso ocorre porque a funcionalidade XIRR calcula os juros ocultos na transação, independentemente de qual lado você esteja na transação.

Limitações com várias soluções

A funcionalidade XIRR do Qlik é definida pela seguinte equação, em que o valor `Rate` é resolvido:

```
XNPV(Rate, pmt, date) = 0
```

Às vezes, é possível que essa equação tenha mais de uma solução. Isso é conhecido como "problema de IRR múltipla" e é causado por um fluxo de caixa não normal (também chamado de fluxo de caixa não convencional). O script de carregamento a seguir mostra um exemplo disso:

```
Cashflow:
LOAD * inline [
Date|Payments
2021-01-01|-200
2022-01-01|500
2023-01-01|-250
] (delimiter is '|');
```

Neste exemplo, há uma solução negativa e uma solução positiva (`Rate` = -0,3 e `Rate` = 0,8). **XIRR()** retornará 0,8.

Quando a funcionalidade XIRR do Qlik procura uma solução, ela começa em `Rate` = 0 e aumenta a taxa em etapas até encontrar uma solução. Se houver mais de uma solução positiva, ela retornará a primeira que encontrar. Se você não conseguir encontrar uma solução positiva, ele redefinirá `Rate` de volta para zero e começará a procurar uma solução na direção negativa.

Observe que é garantido que um fluxo de caixa "normal" tenha apenas uma solução. Fluxo de caixa "normal" significa que todos os pagamentos com o mesmo sinal (positivo ou negativo) estão em um grupo contínuo.

Consulte também:

- *XNPV (page 388)*
- *RangeXIRR (page 1373)*
- Correção e atualização da função XIRR

XIRR - função de gráfico

XIRR() retorna a taxa interna de retorno agregada (anual) para um cronograma de fluxos de caixa (que não é necessariamente periódico) representado por números emparelhados nas expressões especificadas por **pmt** e **date** iterados sobre as dimensões do gráfico. Todos os pagamentos têm descontos baseados em um ano de 365 dias.

Funcionalidade XIRR do Qlik (funções **XIRR()** e **RangeXIRR()**) usa a seguinte equação, resolvendo para o valor `Rate`, para determinar o valor XIRR correto:

```
XNPV(Rate, pmt, date) = 0
```

A equação é resolvida usando uma versão simplificada do método de Newton.

Sintaxe:

```
XIRR([TOTAL [<fld {,fld}>]] pmt, date)
```

Tipo de dados de retorno: numérico

Argumentos

Argumento	Descrição
pmt	Pagamentos. A expressão ou campo que contém os fluxos de caixa correspondentes ao cronograma de pagamentos dado no **date**.
date	A expressão ou campo que contém o cronograma de datas correspondentes aos pagamentos de fluxo de caixa dados em **pmt**.
TOTAL	Se a palavra **TOTAL** ocorrer antes dos argumentos da função, o cálculo será feito sobre todos os valores possíveis, dadas as seleções atuais, e não apenas sobre os pertinentes ao valor dimensional atual, isto é, desconsiderando as dimensões do gráfico. Usando **TOTAL [<fld {.fld}>]**, em que o qualificador **TOTAL** é seguido por uma lista de um ou mais nomes de campo como um subconjunto das variáveis de dimensão de gráfico, você cria um subconjunto dos valores possíveis totais.

Ao trabalhar com essa função, as seguintes limitações são aplicáveis:

- **pmt** e **date** não devem conter funções de agregação, a não ser que essas agregações internas contenham o qualificador **TOTAL**. Para agregações aninhadas mais avançadas, use a função avançada **Aggr** junto com uma dimensão especificada.
- Valores de texto, valores NULLe valores ausentes em qualquer uma das duas partes de um par de dados fazem com que o par de dados inteiro seja ignorado.
- Essa função requer pelo menos um pagamento negativo válido e pelo menos um pagamento positivo válido (com datas válidas correspondentes). Se esses pagamentos não forem fornecidos, um valor NULL será retornado.

Estes tópicos podem ajudar você a trabalhar com essa função:

- *XNPV - função de gráfico (page 394)*: Use essa função para calcular o valor presente líquido agregado para uma programação de fluxos de caixa.
- *RangeXIRR (page 1373)*: **RangeXIRR()** é a função de intervalo equivalente para a função **XIRR()**.

Em diferentes versões do Qlik Sense Client-Managed, há variações no algoritmo subjacente usado por essa função. Para obter mais informações sobre atualizações recentes do algoritmo, consulte o artigo de suporte Correção e atualização da função XIRR.

Exemplo

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo transações de fluxo de caixa.
- Informações armazenadas em uma tabela chamada `Cashflow`.

Script de carregamento

```
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 '|');
```

Resultados

Faça o seguinte:

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione o seguinte cálculo como medida:

```
=XIRR(Payments, Date)
```

Tabela de resultados

=XIRR(Payments, Date)
0.5385

Interpretando o valor de retorno XIRR

A funcionalidade XIRR geralmente é usada para analisar um investimento, em que há um pagamento de saída (negativo) no início e, posteriormente, uma série de pagamentos de receita menores (positivos). Veja a seguir um exemplo simplificado com apenas um pagamento negativo e um positivo:

```
Cashflow:
LOAD * inline [
Date|Payments
2023-01-01|-100
2024-01-01|110
] (delimiter is '|');
```

Fazemos um pagamento inicial de 100 e recebemos 110 de volta depois de exatamente um ano. Isso representa uma taxa de retorno de 10% ao ano. `XIRR(Payments, Date)` retorna um valor de 0,1.

O valor de retorno da funcionalidade XIRR pode ser positivo ou negativo. No caso de um investimento, um resultado negativo indica que o investimento é uma perda. A quantidade de ganho ou perda pode ser calculada com uma simples agregação de soma no campo de pagamentos.

No exemplo acima, estamos emprestando nosso dinheiro por um ano. A taxa de retorno pode ser considerada como juros. Também é possível usar a funcionalidade XIRR quando você está do outro lado da transação (por exemplo, se for o mutuário em vez do credor).

Considere este exemplo:

```
Cashflow:
LOAD * inline [
Date|Payments
2023-01-01|100
2024-01-01|-110
] (delimiter is '|');
```

Isso é igual ao primeiro exemplo, mas invertido. Aqui, tomamos 100 emprestados por um ano e pagamos com juros de 10%. Neste exemplo, o cálculo de XIRR retorna 0,1 (10%), o mesmo valor do primeiro exemplo.

Observe que, no primeiro exemplo, recebemos um lucro de 10 e, no segundo, tivemos uma perda de 10, mas o valor de retorno da funcionalidade XIRR é positivo para ambos os exemplos. Isso ocorre porque a funcionalidade XIRR calcula os juros ocultos na transação, independentemente de qual lado você esteja na transação.

Limitações com várias soluções

A funcionalidade XIRR do Qlik é definida pela seguinte equação, em que o valor `Rate` é resolvido:

```
XNPV(Rate, pmt, date) = 0
```

Às vezes, é possível que essa equação tenha mais de uma solução. Isso é conhecido como "problema de IRR múltipla" e é causado por um fluxo de caixa não normal (também chamado de fluxo de caixa não convencional). O script de carregamento a seguir mostra um exemplo disso:

```
Cashflow:
LOAD * inline [
Date|Payments
2021-01-01|-200
2022-01-01|500
2023-01-01|-250
] (delimiter is '|');
```

Neste exemplo, há uma solução negativa e uma solução positiva (`Rate` = -0,3 e `Rate` = 0,8). **XIRR()** retornará 0,8.

Quando a funcionalidade XIRR do Qlik procura uma solução, ela começa em `Rate` = 0 e aumenta a taxa em etapas até encontrar uma solução. Se houver mais de uma solução positiva, ela retornará a primeira que encontrar. Se você não conseguir encontrar uma solução positiva, ele redefinirá `Rate` de volta para zero e começará a procurar uma solução na direção negativa.

Observe que é garantido que um fluxo de caixa "normal" tenha apenas uma solução. Fluxo de caixa "normal" significa que todos os pagamentos com o mesmo sinal (positivo ou negativo) estão em um grupo contínuo.

Consulte também:

IRR - função de gráfico (page 371)
Aggr - função de gráfico (page 552)
Correção e atualização da função XIRR

XNPV

XNPV() retorna o valor presente líquido agregado para um cronograma de fluxos de caixa (não necessariamente periódicos) representados por números emparelhados em **pmt** e **date**. Todos os pagamentos têm descontos baseados em um ano de 365 dias.

Sintaxe:

```
XNPV(discount_rate, pmt, date)
```

Tipo de dados de retorno: numérico

Por padrão, o resultado será formatado como moeda.

A fórmula para calcular o XNPV é mostrada abaixo:

Fórmula de agregação XNPV

$$XNPV = \sum_{i=1}^{n} \frac{P_i}{(1+rate)^{(di-d1)/365}}$$

na qual:

- P_i = Entradas e saídas líquidas de caixa durante um único período `i`
- d_1 = a data do primeiro pagamento
- d_i = a data do i° pagamento
- `rate` = taxa de desconto

O Valor presente líquido, ou NPV, é usado para calcular o valor total atual de um fluxo futuro de fluxos de caixa, dada uma taxa de desconto. Para calcular XNPV, precisamos estimar fluxos de caixa futuros com datas correspondentes. Depois disso, para cada pagamento, aplicamos a taxa de desconto composta com base na data do pagamento.

Executar a agregação XNPV em uma série de pagamentos é semelhante a executar uma agregação de soma sobre esses pagamentos. A diferença é que cada valor é modificado (ou "descontado") de acordo com a taxa de desconto escolhida (semelhante à taxa de juros) e quanto tempo no futuro está o pagamento. Realizando XNPV com o parâmetro **discount_rate** definido como zero fará com que o XNPV seja equivalente a uma operação de soma (os pagamentos não serão modificados antes de serem somados). Em geral, quanto mais próximo **discount_rate** estiver definido de zero, mais semelhante será o resultado de XNPV ao de uma agregação de soma.

Argumentos

Argumento	Descrição
discount_rate	**discount_rate** é a taxa anual com base na qual os pagamentos devem ser descontados. Um valor de 0,1 indicaria uma taxa de desconto de 10%.
pmt	Pagamentos. A expressão ou campo que contém os fluxos de caixa correspondentes ao cronograma de pagamentos dado no **date**. Valores positivos são considerados entradas, e valores negativos são considerados saídas. ***XNPV()*** *não desconta o fluxo de caixa inicial, pois sempre acontecerá na data de início. Os pagamentos subsequentes são descontados com base em um ano de 365 dias. Ele é diferente de* ***NPV()****, onde também é descontado o primeiro pagamento.*
date	A expressão ou campo que contém o cronograma de datas correspondentes aos pagamentos de fluxo de caixa dados em **pmt**. O primeiro valor é usado como a data inicial para calcular as compensações de tempo para fluxos de caixa futuros.

Ao trabalhar com essa função, as seguintes limitações são aplicáveis:

- Valores de texto, valores NULLe valores ausentes em qualquer uma das duas partes de um par de dados fazem com que o par de dados inteiro seja ignorado.

Quando usar

- `XNPV()` é usada na modelagem financeira para calcular o valor presente líquido (NPV) de uma oportunidade de investimento.
- Devido à sua maior precisão, XNPV é preferível a NPV para todos os tipos de modelos financeiros.

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplo 1: Pagamento único (script)

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados de um projeto e seu fluxo de caixa por um ano, em uma tabela denominada `CashFlow`. A data inicial para o cálculo está definida como 1º de julho de 2022, com um fluxo de caixa líquido de 0. Depois de um ano, ocorre um fluxo de caixa de $ 1.000.
- Um carregamento residente da tabela `CashFlow`, que é usado para calcular o campo `XNPV` do projeto em uma tabela denominada `XNPV`.
- Uma taxa de desconto codificada de 10% (0,1) é usada no cálculo de XNPV.
- Uma instrução `Group By` é usada para agrupar todos os pagamentos do projeto.

Script de carregamento

```
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;
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `PrjId`
- `XNPV`

Tabela de resultados

PrjId	XNPV
1	$909.09

De acordo com a fórmula, o valor de XNPV para o primeiro registro é 0 e, para o segundo registro, o valor de XNPV é $ 909,09. Portanto, o XNPV total é $ 909,09.

Exemplo 2: Vários pagamentos (script)

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados de um projeto e seu fluxo de caixa por um ano, em uma tabela denominada `CashFlow`.
- Um carregamento residente da tabela `CashFlow`, que é usado para calcular o campo `XNPV` do projeto em uma tabela denominada `XNPV`.
- Uma taxa de desconto codificada de 10% (0,1) é usada no cálculo de XNPV.
- Uma instrução `Group By` é usada para agrupar todos os pagamentos do projeto.

Script de carregamento

```
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;
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `PrjId`
- `XNPV`

Tabela de resultados

PrjId	XNPV
1	$1322.21

Neste exemplo, um pagamento de $ 1.000 é recebido no final do primeiro ano e um pagamento de $ 500 é recebido no final do segundo ano. Com uma taxa de desconto de 10% por período, o XNPV efetivo é igual a $ 1.322,21.

Observe que somente a primeira linha de dados deve se referir à data base para os cálculos. Para o restante das linhas, a ordem não é importante, pois o parâmetro de data é usado para calcular o período decorrido.

Exemplo 3: Vários pagamentos e fluxos de caixa irregulares (script)

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Taxas de desconto para dois projetos em uma tabela denominada `Project`.
- Fluxos de caixa para vários períodos para cada projeto por ID e datas de projeto. O campo `Dates` é usado para calcular a duração pela qual a taxa de desconto é aplicada ao fluxo de caixa. Além do primeiro registro (fluxo de caixa inicial e data), a ordem dos registros não é importante, e alterá-la não afetará os cálculos.
- Usando uma combinação de `NoConcatenate`, carregamentos residentes e a função `Left Join`, é criada uma tabela temporária `tmpNPV` que combina os registros das tabelas `Project` e `CashFlow` em uma tabela simples. Essa tabela terá taxas de desconto repetidas para cada fluxo de caixa.
- Um carregamento residente da tabela `tmpNPV`, que é usado para calcular o campo `XNPV` de cada projeto em uma tabela denominada `XNPV`.
- A taxa de desconto de valor único associada a cada projeto é obtida usando a função `only()` e é usada no cálculo de `XNPV` de cada projeto.
- Uma instrução `Group By`, agrupada por ID do projeto, é usada para agrupar todos os pagamentos e as datas correspondentes de cada projeto.
- Para evitar que dados sintéticos ou redundantes sejam carregados no modelo de dados, a tabela `tmpXNPV` é descartada no final do script.

Script de carregamento

```
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;
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- PrjId
- XNPV

Tabela de resultados

PrjId	XNPV
1	$1735.54
2	$278.36

O ID de projeto 1 tem um fluxo de caixa inicial de $ 0 em 1º de julho de 2021. Há dois pagamentos de $ 1.000 a serem recebidos no final de dois anos subsequentes, com uma taxa de desconto de 10% por período. Portanto, o XNPV efetivo é igual a $ 1.735,54.

O ID de projeto 2 tem uma saída inicial de $ 1.000 (portanto, o sinal negativo) em 1º de julho de 2020. Depois de dois anos, espera-se um pagamento de $ 500. Após 3 anos, outro pagamento de $ 500 é esperado. Finalmente, em 1º de julho de 2024, espera-se um pagamento de $ 1.000. Com a taxa de desconto de 15%, o XNPV efetivo é igual a $ 278,36.

Consulte também:

Drop table (page 152)

group by (page 162)
Join (page 73)
Max (page 336)
NoConcatenate (page 92)
NPV - função de gráfico (page 380)
Only (page 346)

XNPV - função de gráfico

XNPV() retorna o valor líquido atual agregado de uma programação de fluxos de caixa (não necessariamente periódica) representados por números emparelhados nas expressões dadas por **pmt** e **date** repetidos nas dimensões de gráfico. Todos os pagamentos têm descontos baseados em um ano de 365 dias.

Sintaxe:

```
XNPV([TOTAL [<fld{,fld}>]] discount_rate, pmt, date)
```

Tipo de dados de retorno: numérico

Por padrão, o resultado será formatado como moeda.

A fórmula para calcular o XNPV é mostrada abaixo:

Fórmula de agregação XNPV

$$XNPV = \sum_{i=1}^{n} \frac{P_i}{(1+rate)^{(di-d1)/365}}$$

na qual:

- P_i = Entradas e saídas líquidas de caixa durante um único período i
- d_1 = a data do primeiro pagamento
- d_i = a data do i° pagamento
- `rate` = taxa de desconto

O Valor presente líquido, ou NPV, é usado para calcular o valor total atual de um fluxo futuro de fluxos de caixa, dada uma taxa de desconto. Para calcular XNPV, precisamos estimar fluxos de caixa futuros com datas correspondentes. Depois disso, para cada pagamento, aplicamos a taxa de desconto composta com base na data do pagamento.

Executar a agregação XNPV em uma série de pagamentos é semelhante a executar uma agregação de soma sobre esses pagamentos. A diferença é que cada valor é modificado (ou "descontado") de acordo com a taxa de desconto escolhida (semelhante à taxa de juros) e quanto tempo no futuro está o pagamento. Realizando XNPV com o parâmetro **discount_rate** definido como zero fará com que o XNPV seja equivalente a uma operação de soma (os pagamentos não serão modificados antes de serem somados). Em geral, quanto mais próximo **discount_rate** estiver definido de zero, mais semelhante será o resultado de XNPV ao de uma agregação de soma.

Argumentos

Argumento	Descrição
discount_rate	**discount_rate** é a taxa anual com base na qual os pagamentos devem ser descontados. Um valor de 0,1 indicaria uma taxa de desconto de 10%.
pmt	Pagamentos. A expressão ou campo que contém os fluxos de caixa correspondentes ao cronograma de pagamentos dado no **date**. Valores positivos são considerados entradas, e valores negativos são considerados saídas. *__XNPV()__ não desconta o fluxo de caixa inicial, pois sempre acontecerá na data de início. Os pagamentos subsequentes são descontados com base em um ano de 365 dias. Ele é diferente de __NPV()__, onde também é descontado o primeiro pagamento.*
date	A expressão ou campo que contém o cronograma de datas correspondentes aos pagamentos de fluxo de caixa dados em **pmt**. O primeiro valor é usado como a data inicial para calcular as compensações de tempo para fluxos de caixa futuros.
TOTAL	Se a palavra **TOTAL** ocorrer antes dos argumentos da função, o cálculo será feito sobre todos os valores possíveis, dadas as seleções atuais, e não apenas sobre os pertinentes ao valor dimensional atual, isto é, desconsiderando as dimensões do gráfico. Usando **TOTAL [<fld {.fld}>]**, em que o qualificador **TOTAL** é seguido por uma lista de um ou mais nomes de campo como um subconjunto das variáveis de dimensão de gráfico, você cria um subconjunto dos valores possíveis totais.

Ao trabalhar com essa função, as seguintes limitações são aplicáveis:

- **discount_rate**,**pmt** e **date** não devem conter funções de agregação, a não ser que essas agregações internas contenham os qualificadores **TOTAL** ou **ALL**. Para agregações aninhadas mais avançadas, use a função avançada **Aggr** junto com uma dimensão especificada.
- Valores de texto, valores NULLe valores ausentes em qualquer uma das duas partes de um par de dados fazem com que o par de dados inteiro seja ignorado.

Quando usar

- `XNPV()` é usada na modelagem financeira para calcular o valor presente líquido (NPV) de uma oportunidade de investimento.
- Devido à sua maior precisão, XNPV é preferível a NPV para todos os tipos de modelos financeiros.

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplo

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo transações de fluxo de caixa.
- Informações armazenadas em uma tabela chamada `Cashflow`.

Script de carregamento

```
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 '|');
```

Resultados

Faça o seguinte:

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione o seguinte cálculo como medida:

```
=XNPV(0.09, Payments, Date)
```

Tabela de resultados

=XNPV(0.09, Payments, Date)
$3062.49

Consulte também:

NPV - função de gráfico (page 380)
Aggr - função de gráfico (page 552)

Funções de agregação estatística

Cada função é descrita adicionalmente após a visão geral. Você também pode clicar no nome da função na sintaxe para acessar imediatamente os detalhes dessa função específica.

Funções de agregação estatística no script de carga de dados

As seguintes funções de agregação estatística podem ser usadas em scripts.

Avg

Avg() encontra o valor médio dos dados agregados na expressão sobre um número de registros definidos por uma cláusula **group by**.

```
Avg  ([distinct] expression)
```

Correl

Correl() retorna o coeficiente de correlação agregada de uma série de coordenadas representadas por números em x-expression e y-expression iteradas em um número de registros, conforme definido por uma cláusula **group by**.

```
Correl (x-expression, y-expression)
```

Fractile

Fractile() encontra o valor que corresponde ao fractil inclusivo (quantil) dos dados agregados na expressão sobre um número de registros definidos por uma cláusula **group by**.

```
Fractile (expression, fractile)
```

FractileExc

FractileExc() encontra o valor que corresponde ao fractil exclusivo (quantil) dos dados agregados na expressão sobre um número de registros definidos por uma cláusula **group by**.

```
FractileExc (expression, fractile)
```

Kurtosis

Kurtosis() retorna a contagem da expressão sobre um número de registros definidos por uma cláusula **group by**.

```
Kurtosis  ([distinct ] expression )
```

LINEST_B

LINEST_B() retorna o valor b (intercepção y) agregado de uma regressão linear definida pela equação y=mx+b para uma série de coordenadas representadas por números pareados em x-expression e y-expression iteradas em um número de registros, conforme definido por uma cláusula **group by**.

```
LINEST_B (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_df

LINEST_DF() retorna os graus de liberdade agregados de uma regressão linear definida pela equação y=mx+b para uma série de coordenadas representadas por números pareados em x-expression e y-expression iteradas em um número de registros, conforme definido por uma cláusula **group by**.

```
LINEST_DF  (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_f

Essa função de script retorna a estatística F agregada ($r^2/(1-r^2)$) de uma regressão linear definida pela equação y=mx+b para uma série de coordenadas representadas por números pareados em x-expression e y-expression iteradas em um número de registros, conforme definido por uma cláusula **group by**.

```
LINEST_F (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_m

LINEST_M() retorna o valor m (declive) agregado de uma regressão linear definida pela equação y=mx+b para uma série de coordenadas representadas por números pareados em x-expression e y-expression iteradas em um número de registros, conforme definido por uma cláusula **group by**.

```
LINEST_M  (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_r2

LINEST_R2() retorna o valor agregado r^2 (coeficiente de determinação) de uma regressão linear definida pela equação y=mx+b para uma série de coordenadas representadas por números pareados em x-expression e y-expression iteradas em um número de registros, conforme definido por uma cláusula **group by**.

```
LINEST_R2 (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_seb

LINEST_SEB() retorna o erro padrão agregado do valor b de uma regressão linear definida pela equação y=mx+b para uma série de coordenadas representadas por números pareados em x-expression e y-expression iteradas em um número de registros, conforme definido por uma cláusula **group by**.

```
LINEST_SEB  (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_sem

LINEST_SEM() retorna o erro padrão agregado do valor m de uma regressão linear definida pela equação y=mx+b para uma série de coordenadas representadas por números pareados em x-expression e y-expression iteradas em um número de registros, conforme definido por uma cláusula **group by**.

```
LINEST_SEM  (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_sey

LINEST_SEY() retorna o erro padrão agregado da estimativa y de uma regressão linear definida pela equação y=mx+b para uma série de coordenadas representadas por números pareados em x-expression e y-expression iteradas em um número de registros, conforme definido por uma cláusula **group by**.

```
LINEST_SEY  (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_ssreg

LINEST_SSREG() retorna a soma de regressão agregada dos quadrados de uma regressão linear definida pela equação y=mx+b para uma série de coordenadas representadas por números pareados em x-expression e y-expression iteradas em um número de registros, conforme definido por uma cláusula **group by**.

```
LINEST_SSREG  (y-expression, x-expression [, y0 [, x0 ]])
```

Linest_ssresid

LINEST_SSRESID() retorna a soma residual agregada dos quadrados de uma regressão linear definida pela equação y=mx+b para uma série de coordenadas representadas por números pareados em x-expression e y-expression iteradas em um número de registros, conforme definido por uma cláusula **group by**.

```
LINEST_SSRESID  (y-expression, x-expression [, y0 [, x0 ]])
```

Median

Median() retorna a mediana agregada dos valores na expressão sobre um número de registros como definido por uma cláusula **group by**.

```
Median (expression)
```

Skew

Skew() retorna a assimetria da expressão sobre um número de registro definido por uma cláusula **group by**.

```
Skew ([ distinct] expression)
```

Stdev

Stdev() retorna o desvio padrão dos valores dados pela expressão sobre um número de registro conforme definido por uma cláusula **group by**.

```
Stdev ([distinct] expression)
```

Sterr

Sterr() retorna o erro padrão agregado (stdev/sqrt(n)) de uma série de valores representados pela expressão iterada sobre um número de registros, conforme definido por uma cláusula **group by**.

```
Sterr  ([distinct] expression)
```

STEYX

STEYX() retorna o erro padrão agregado do valor y previsto para cada valor x na regressão de uma série de coordenadas representada por números pareados em x-expression e y-expression , com iterações em vários registros definidos por uma cláusula **group by**.

```
STEYX  (y-expression, x-expression)
```

Funções de agregação estatística em expressões de gráfico

As seguintes funções de agregação estatística podem ser usadas em gráficos:

Avg

Avg() retorna a média agregada da expressão ou campo repetida nas dimensões de gráfico.

```
Avg - função de gráfico({[SetExpression] [DISTINCT] [TOTAL [<fld{, fld}>]]}
expr)
```

Correl

Correl() retorna o coeficiente da correlação agregado para dois conjuntos de dados. A função de correlação é uma medida da relação entre os conjuntos de dados e é agregada para pares de valores (x,y) iterados nas dimensões de gráfico.

```
Correl - função de gráfico({[SetExpression] [TOTAL [<fld {, fld}>]]} value1,
value2 )
```

Fractile

Fractile() encontra o valor que corresponde ao fractil inclusivo (quantil) dos dados agregados no intervalo dado pela expressão repetida nas dimensões de gráfico.

```
Fractile - função de gráfico({[SetExpression] [TOTAL [<fld {, fld}>]]} expr,
fraction)
```

FractileExc

FractileExc() encontra o valor que corresponde ao fractil exclusivo (quantil) dos dados agregados no intervalo dado pela expressão repetida nas dimensões de gráfico.

```
FractileExc - função de gráfico({[SetExpression] [TOTAL [<fld {, fld}>]]}
expr, fraction)
```

Kurtosis

Kurtosis() encontra a curtose do intervalo de dados agregados na expressão ou campo repetidos nas dimensões de gráfico.

```
Kurtosis - função de gráfico({[SetExpression] [DISTINCT] [TOTAL [<fld{,
fld}>]]} expr)
```

LINEST_b

LINEST_B() retorna o valor agregado b (intercepção y) de uma regressão linear definida pela equação y=mx+b de uma série de coordenadas representadas por números emparelhados nas expressões dadas pelas expressões **x_value** e **y_value**, repetidas nas dimensões de gráfico.

```
LINEST_R2 - função de gráfico({[SetExpression] [TOTAL [<fld{ ,fld}>]] }y_
value, x_value[, y0_const[, x0_const]])
```

LINEST_df

LINEST_DF() retorna os graus de liberdade agregados de uma regressão linear definida pela equação y=mx+b de uma série de coordenadas representadas por números emparelhados nas expressões dadas por **x_value** e **y_value**, repetidos nas dimensões de gráfico.

```
LINEST_DF - função de gráfico({[SetExpression] [TOTAL [<fld{, fld}>]]} y_
value, x_value [, y0_const [, x0_const]])
```

LINEST_f

LINEST_F() retorna a estatística F agregada (r2/(1-r2)) de uma regressão linear definida pela equação y=mx+b de uma série de coordenadas representadas por números emparelhados nas expressões dadas por **x_value** e **y_value**, repetidos nas dimensões de gráfico.

```
LINEST_F - função de gráfico({[SetExpression] [TOTAL[<fld{, fld}>]]} y_value,
x_value [, y0_const [, x0_const]])
```

LINEST_m

LINEST_M() retorna o valor agregado m (declive) de uma regressão linear definida pela equação y=mx+b de uma série de coordenadas representadas por números emparelhados dados pelas expressões **x_value** e **y_value**, repetidos nas dimensões de gráfico.

```
LINEST_M - função de gráfico({[SetExpression] [TOTAL[<fld{, fld}>]]} y_value,
x_value [, y0_const [, x0_const]])
```

LINEST_r2

LINEST_R2() retorna o valor r2 agregado (coeficiente de determinação) de uma regressão linear definida pela equação y=mx+b de uma série de coordenadas representadas por números emparelhados dados pelas expressões **x_value** e **y_value**, repetidos nas dimensões de gráfico.

```
LINEST_R2 - função de gráfico({[SetExpression] [TOTAL [<fld{ ,fld}>]] }y_
value, x_value[, y0_const[, x0_const]])
```

LINEST_seb

LINEST_SEB() retorna o erro padrão agregado do valor b de uma regressão linear definida pela equação y=mx+b de uma série de coordenadas representadas por números emparelhados dados pelas expressões **x_value** e **y_value**, repetidos nas dimensões de gráfico.

```
LINEST_SEB - função de gráfico({[SetExpression] [TOTAL [<fld{ ,fld}>]] }y_
value, x_value[, y0_const[, x0_const]])
```

LINEST_sem

LINEST_SEM() retorna o erro padrão agregado do valor m de uma regressão linear definida pela equação y=mx+b de uma série de coordenadas representadas por números emparelhados dados pelas expressões **x_value** e **y_value**, repetidos nas dimensões de gráfico.

```
LINEST_SEM - função de gráfico([{set_expression}][ distinct ] [total [<fld
{,fld}>] ] y-expression, x-expression [, y0 [, x0 ]] )
```

LINEST_sey

LINEST_SEY() retorna o erro padrão agregado da estimativa y de uma regressão linear definida pela equação y=mx+b de uma série de coordenadas representadas por números emparelhados dados pelas expressões **x_value** e **y_value**, repetidos nas dimensões de gráfico.

```
LINEST_SEY - função de gráfico({[SetExpression] [TOTAL [<fld{ ,fld}>]] }y_
value, x_value[, y0_const[, x0_const]])
```

LINEST_ssreg

LINEST_SSREG() retorna a soma de regressão agregada de quadrados de uma regressão linear definida pela equação y=mx+b de uma série de coordenadas representadas por números emparelhados dados pelas expressões **x_value** e **y_value**, repetidos nas dimensões de gráfico.

```
LINEST_SSREG - função de gráfico({[SetExpression] [TOTAL [<fld{ ,fld}>]] }y_
value, x_value[, y0_const[, x0_const]])
```

LINEST_ssresid

LINEST_SSRESID() retorna a soma residual agregada dos quadrados de uma regressão linear definida pela equação y=mx+b de uma série de coordenadas representadas por números emparelhados nas expressões dadas por **x_value** e **y_value**, repetidos nas dimensões de gráfico.

```
LINEST_SSRESID - função de gráficoLINEST_SSRESID() retorna a soma residual
agregada dos quadrados de uma regressão linear definida pela equação y=mx+b
de uma série de coordenadas representadas por números emparelhados nas
expressões dadas por x_value e y_value, repetidos nas dimensões de gráfico.
LINEST_SSRESID([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value,
x_value[, y0_const[, x0_const]]) numérico ArgumentosArgumentoDescriçãoy_
valueA expressão ou campo que contém o intervalo de valores y conhecidos a
ser medido.x_valueA expressão ou campo que contém o intervalo de valores x
conhecidos a ser medido.y0, x0Um valor opcional y0 pode ser declarado,
forçando a linha de regressão a passar pelo eixo y em um determinado ponto.
Declarando y0 e x0, é possível forçar a linha de regressão a passar por uma
única coordenada fixa. A menos que y0 e x0 sejam declarados, a função exigirá
pelo menos dois pares de dados válidos para efetuar o cálculo. Se y0 e x0
forem declarados, será necessário um único par de dados. SetExpressionPor
padrão, a função de agregação agregará um conjunto de registros possíveis
definidos pela seleção. Um conjunto de registros alternativos pode ser
definido por uma expressão de análise. DISTINCTSe a palavra DISTINCT aparecer
antes dos argumentos de função, as duplicatas resultantes da avaliação dos
argumentos de função serão ignoradas. TOTALSe a palavra TOTAL ocorrer antes
dos argumentos da função, o cálculo será feito sobre todos os valores
possíveis, dadas as seleções atuais, e não apenas sobre os pertinentes ao
valor dimensional atual, isto é, desconsiderando as dimensões do gráfico.
Usando TOTAL [<fld {.fld}>], em que o qualificador TOTAL é seguido por uma
lista de um ou mais nomes de campo como um subconjunto das variáveis de
dimensão de gráfico, você cria um subconjunto dos valores possíveis totais.Um
valor opcional y0 pode ser declarado, forçando a linha de regressão a passar
pelo eixo y em um determinado ponto. Declarando y0 e x0, é possível forçar a
linha de regressão a passar por uma única coordenada fixa.  O parâmetro da
função de agregação não deve conter outras funções de agregação, a menos que
essas agregações internas contenham o qualificador TOTAL. Para agregações
aninhadas mais avançadas, use a função avançada Aggr junto com uma dimensão
especificada. Valores de texto, valores NULLe valores ausentes em qualquer
```

```
uma das duas partes de um par de dados fazem com que o par de dados inteiro
seja ignorado.  An example of how to use linest functionsavg({[SetExpression]
[TOTAL [<fld{ ,fld}>]] }y_value, x_value[, y0_const[, x0_const]])
```

Median

Median() retorna o valor médio do intervalo de valores agregados na expressão repetida nas dimensões de gráfico.

```
Median - função de gráfico({[SetExpression] [TOTAL [<fld{, fld}>]]} expr)
```

MutualInfo

MutualInfo calcula as informações mútuas (MI) entre dois campos ou entre valores agregados em **Aggr()**.

```
MutualInfo - função de gráfico{[SetExpression] [DISTINCT] [TOTAL target,
driver [, datatype [, breakdownbyvalue [, samplesize ]]])
```

Skew

Skew() retorna a assimetria agregada da expressão ou campo repetida nas dimensões de gráfico.

```
Skew - função de gráfico{[SetExpression] [DISTINCT] [TOTAL [<fld{ ,fld}>]]}
expr)
```

Stdev

Stdev() encontra o desvio padrão do intervalo de dados agregados na expressão ou campo repetidos nas dimensões de gráfico.

```
Stdev - função de gráfico({[SetExpression] [DISTINCT] [TOTAL [<fld{, fld}>]]}
expr)
```

Sterr

Sterr() encontra o valor do erro padrão da média, (stdev/sqrt(n)), da série de valores agregados na expressão repetida nas dimensões de gráfico.

```
Sterr - função de gráfico({[SetExpression] [DISTINCT] [TOTAL[<fld{, fld}>]]}
expr)
```

STEYX

STEYX() retorna o erro padrão agregado ao prever os valores y de cada valor x em uma regressão linear dada por uma série de coordenadas representadas por números emparelhados nas expressões dadas pelos **y_value** e **x_value**.

```
STEYX - função de gráfico{[SetExpression] [TOTAL [<fld{, fld}>]]} y_value, x_
value)
```

Avg

Avg() encontra o valor médio dos dados agregados na expressão sobre um número de registros definidos por uma cláusula **group by**.

Sintaxe:

```
Avg([DISTINCT] expr)
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
expr	A expressão ou campo que contém os dados a serem medidos.
DISTINCT	Se a palavra **distinct** aparecer antes da expressão, todas as duplicatas serão ignoradas.

Exemplos e resultados:

Adicione o script de exemplo ao seu aplicativo e execute-o. Para ver o resultado, adicione os campos listados na coluna de resultados a uma pasta no seu aplicativo.

Dados resultantes

Exemplo	Resultado

`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`
	Isso pode ser verificado na pasta ao criar uma tabela incluindo a medida.
	`Sum(Sales)/12`

Dado que a tabela **Temp** é carregada como no exemplo anterior:
 	
`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`
	Apenas os valores distintos são contados. Divida o total pelo número de valores não duplicados.

Avg - função de gráfico

Avg() retorna a média agregada da expressão ou campo repetida nas dimensões de gráfico.

Sintaxe:

```
Avg([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr)
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
expr	A expressão ou campo que contém os dados a serem medidos.
SetExpression	Por padrão, a função de agregação agregará um conjunto de registros possíveis definidos pela seleção. Um conjunto de registros alternativos pode ser definido por uma expressão de análise.
DISTINCT	Se a palavra **DISTINCT** aparecer antes dos argumentos de função, as duplicatas resultantes da avaliação dos argumentos de função serão ignoradas.
TOTAL	Se a palavra **TOTAL** ocorrer antes dos argumentos da função, o cálculo será feito sobre todos os valores possíveis, dadas as seleções atuais, e não apenas sobre os pertinentes ao valor dimensional atual, isto é, desconsiderando as dimensões do gráfico. Usando **TOTAL [<fld {.fld}>]**, em que o qualificador **TOTAL** é seguido por uma lista de um ou mais nomes de campo como um subconjunto das variáveis de dimensão de gráfico, você cria um subconjunto dos valores possíveis totais.

Limitações:

O parâmetro da função de agregação não deve conter outras funções de agregação, a menos que essas agregações internas contenham o qualificador **TOTAL**. Para agregações aninhadas mais avançadas, use a função avançada **Aggr** junto com uma dimensão especificada.

Exemplos e resultados:

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

Exemplos de funções

Exemplo	Resultado
Avg(Sales)	Para uma tabela incluindo a dimensão Customer e a medição Avg([Sales]), se **Totals** for mostrado, o resultado será 2566.
Avg([TOTAL (Sales))	53.458333 para todos os valores de Customer, porque o qualificador TOTAL significa que as dimensões são desconsideradas.
Avg(DISTINCT (Sales))	51,862069 para o total, porque usar o qualificador Distinct significa que apenas valores únicos em Sales para cada Customer são avaliados.

Dados usados nos exemplos:

```
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 '|');
```

Consulte também:

Aggr - função de gráfico (page 552)

Correl

Correl() retorna o coeficiente de correlação agregada de uma série de coordenadas representadas por números em x-expression e y-expression iteradas em um número de registros, conforme definido por uma cláusula **group by**.

Sintaxe:

```
Correl(value1, value2)
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
value1, value2	As expressões ou campos contendo os dois conjuntos de amostra para o qual o coeficiente de correlação deve ser medido.

Limitações:

Valores de texto, valores NULLe valores ausentes em qualquer uma das duas partes de um par de dados fazem com que o par de dados inteiro seja ignorado.

Exemplos e resultados:

Adicione o script de exemplo ao seu aplicativo e execute-o. Para ver o resultado, adicione os campos listados na coluna de resultados a uma pasta no seu aplicativo.

Dados resultantes

Exemplo	Resultado
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;	Em uma tabela com a dimensão `Correl_Salary`, o resultado do cálculo Correl() no script de carregamento de dados será: 0.9270611

Correl - função de gráfico

Correl() retorna o coeficiente da correlação agregado para dois conjuntos de dados. A função de correlação é uma medida da relação entre os conjuntos de dados e é agregada para pares de valores (x,y) iterados nas dimensões de gráfico.

Sintaxe:

```
Correl([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] value1, value2 )
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
value1, value2	As expressões ou campos contendo os dois conjuntos de amostra para o qual o coeficiente de correlação deve ser medido.
SetExpression	Por padrão, a função de agregação agregará um conjunto de registros possíveis definidos pela seleção. Um conjunto de registros alternativos pode ser definido por uma expressão de análise.
DISTINCT	Se a palavra **DISTINCT** aparecer antes dos argumentos de função, as duplicatas resultantes da avaliação dos argumentos de função serão ignoradas.
TOTAL	Se a palavra **TOTAL** ocorrer antes dos argumentos da função, o cálculo será feito sobre todos os valores possíveis, dadas as seleções atuais, e não apenas sobre os pertinentes ao valor dimensional atual, isto é, desconsiderando as dimensões do gráfico. Usando **TOTAL [<fld {.fld}>]**, em que o qualificador **TOTAL** é seguido por uma lista de um ou mais nomes de campo como um subconjunto das variáveis de dimensão de gráfico, você cria um subconjunto dos valores possíveis totais.

Limitações:

O parâmetro da função de agregação não deve conter outras funções de agregação, a menos que essas agregações internas contenham o qualificador **TOTAL**. Para agregações aninhadas mais avançadas, use a função avançada **Aggr** junto com uma dimensão especificada.

Valores de texto, valores NULLe valores ausentes em qualquer uma das duas partes de um par de dados fazem com que o par de dados inteiro seja ignorado.

Exemplos e resultados:

Exemplos de funções

Exemplo	Resultado
`Correl (Age, Salary)`	Para uma tabela incluindo a dimensão `Employee name` e a medição `Correl(Age, Salary)`, o resultado é 0,9270611. O resultado é exibido apenas para a célula de totais.
`Correl (TOTAL Age, Salary))`	0.927. Este e os resultados a seguir são mostrados em três casas decimais para fins de legibilidade. Se você criar uma caixa de filtro com a dimensão Gender e fizer seleções a partir dela, verá o resultado 0,951 quando Female for selecionado e 0,939, se Male for selecionado. Isso ocorre porque a seleção exclui todos os resultados que não pertencem a outro valor de Gender.

Exemplo	Resultado
`Correl({1} TOTAL Age, Salary))`	0.927. Independente das seleções. Isso ocorre porque a expressão de conjunto {1} desconsidera todas as seleções e dimensões.
`Correl (TOTAL <Gender> Age, Salary))`	0,927 na célula de totais, 0,939 para todos os valores de Male e 0,951 para todos os valores de Female. Isso corresponde aos resultados de fazer as seleções em uma caixa de filtro com base em Gender.

Dados usados nos exemplos:

```
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 '|');
```

Consulte também:

- *Aggr - função de gráfico (page 552)*
- *Avg - função de gráfico (page 404)*
- *RangeCorrel (page 1343)*

Fractile

Fractile() encontra o valor que corresponde ao fractil inclusivo (quantil) dos dados agregados na expressão sobre um número de registros definidos por uma cláusula **group by**.

Você pode usar FractileExc (page 414) para calcular o fractil exclusivo.

Sintaxe:

```
Fractile(expr, fraction)
```

Tipo de dados de retorno: numérico

A função retorna o valor correspondente à classificação, conforme definido por `rank = fraction * (N-1) + 1`, em que `N` é o número de valores em `expr`. Se `rank` for um número não inteiro, uma interpolação será feita entre os dois valores mais próximos.

Argumentos:

Argumentos

Argumento	Descrição
expr	A expressão ou campo que contém os dados a serem usados ao calcular o fractil.
fraction	Um número entre 0 e 1 correspondente ao fractil (quantil expressado como uma fração) que será calculado.

Exemplos e resultados:

Adicione o script de exemplo ao seu aplicativo e execute-o. Para ver o resultado, adicione os campos listados na coluna de resultados a uma pasta no seu aplicativo.

Dados resultantes

Exemplo	Resultado
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;	Na tabela com as dimensões Type, MyFractile e Fractile() os resultados dos calculados no script de carregamento de dados são: Type MyFractile Comparison 27.5 Observation 36

Fractile - função de gráfico

Fractile() encontra o valor que corresponde ao fractil inclusivo (quantil) dos dados agregados no intervalo dado pela expressão repetida nas dimensões de gráfico.

Você pode usar FractileExc - função de gráfico (page 416) para calcular o fractil exclusivo.

Sintaxe:

```
Fractile([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr, fraction)
```

Tipo de dados de retorno: numérico

A função retorna o valor correspondente à classificação, conforme definido por `rank = fraction * (N-1) + 1`, em que `N` é o número de valores em `expr`. Se `rank` for um número não inteiro, uma interpolação será feita entre os dois valores mais próximos.

Argumentos:

Argumentos

Argumento	Descrição
expr	A expressão ou campo que contém os dados a serem usados ao calcular o fractil.
fraction	Um número entre 0 e 1 correspondente ao fractil (quantil expressado como uma fração) que será calculado.
SetExpression	Por padrão, a função de agregação agregará um conjunto de registros possíveis definidos pela seleção. Um conjunto de registros alternativos pode ser definido por uma expressão de análise.
DISTINCT	Se a palavra **DISTINCT** aparecer antes dos argumentos de função, as duplicatas resultantes da avaliação dos argumentos de função serão ignoradas.
TOTAL	Se a palavra **TOTAL** ocorrer antes dos argumentos da função, o cálculo será feito sobre todos os valores possíveis, dadas as seleções atuais, e não apenas sobre os pertinentes ao valor dimensional atual, isto é, desconsiderando as dimensões do gráfico. Usando **TOTAL [<fld {.fld}>]**, em que o qualificador **TOTAL** é seguido por uma lista de um ou mais nomes de campo como um subconjunto das variáveis de dimensão de gráfico, você cria um subconjunto dos valores possíveis totais.

Limitações:

O parâmetro da função de agregação não deve conter outras funções de agregação, a menos que essas agregações internas contenham o qualificador **TOTAL**. Para agregações aninhadas mais avançadas, use a função avançada **Aggr** junto com uma dimensão especificada.

Exemplos e resultados:

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

Exemplos de funções

Exemplo	Resultado
`Fractile (Sales, 0.75)`	Para uma tabela incluindo a dimensão `Customer` e a medição `Fractile([Sales])`, se **Totals** for mostrado, o resultado será 71,75. Esse é o ponto da distribuição de valores de `Sales` do qual 75% dos valores ficam abaixo.

Exemplo	Resultado
`Fractile (TOTAL Sales, 0.75))`	71.75 para todos os valores de `Customer`, porque o qualificador TOTAL significa que as dimensões são desconsideradas.
`Fractile (DISTINCT Sales, 0.75)`	70 para o total, porque usar o qualificador DISTINCT significa que apenas valores únicos em `Sales` para cada `Customer` são avaliados.

Dados usados nos exemplos:

```
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 '|');
```

Consulte também:

Aggr - função de gráfico (page 552)

FractileExc

FractileExc() encontra o valor que corresponde ao fractil exclusivo (quantil) dos dados agregados na expressão sobre um número de registros definidos por uma cláusula **group by**.

Você pode usar Fractile (page 410) para calcular o fractil inclusivo.

Sintaxe:

```
FractileExc(expr, fraction)
```

Tipo de dados de retorno: numérico

A função retorna o valor correspondente à classificação, conforme definido por `rank = fraction * (N+1)`, em que `N` é o número de valores em `expr`. Se `rank` for um número não inteiro, uma interpolação será feita entre os dois valores mais próximos.

Argumentos:

Argumentos

Argumento	Descrição
expr	A expressão ou campo que contém os dados a serem usados ao calcular o fractil.
fraction	Um número entre 0 e 1 correspondente ao fractil (quantil expressado como uma fração) que será calculado.

Exemplos e resultados:

Adicione o script de exemplo ao seu aplicativo e execute-o. Para ver o resultado, adicione os campos listados na coluna de resultados a uma pasta no seu aplicativo.

Dados resultantes

Exemplo	Resultado
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;	Na tabela com as dimensões Type, MyFractile e FractileExc() os resultados dos calculados no script de carregamento de dados são: Type MyFractile Comparison 28.5 Observation 38

FractileExc - função de gráfico

FractileExc() encontra o valor que corresponde ao fractil exclusivo (quantil) dos dados agregados no intervalo dado pela expressão repetida nas dimensões de gráfico.

Você pode usar Fractile - função de gráfico (page 412) para calcular o fractil inclusivo.

Sintaxe:

```
FractileExc([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr,
fraction)
```

Tipo de dados de retorno: numérico

A função retorna o valor correspondente à classificação, conforme definido por `rank = fraction * (N+1)`, em que `N` é o número de valores em `expr`. Se `rank` for um número não inteiro, uma interpolação será feita entre os dois valores mais próximos.

Argumentos:

Argumentos

Argumento	Descrição
expr	A expressão ou campo que contém os dados a serem usados ao calcular o fractil.
fraction	Um número entre 0 e 1 correspondente ao fractil (quantil expressado como uma fração) que será calculado.
SetExpression	Por padrão, a função de agregação agregará um conjunto de registros possíveis definidos pela seleção. Um conjunto de registros alternativos pode ser definido por uma expressão de análise.
DISTINCT	Se a palavra **DISTINCT** aparecer antes dos argumentos de função, as duplicatas resultantes da avaliação dos argumentos de função serão ignoradas.
TOTAL	Se a palavra **TOTAL** ocorrer antes dos argumentos da função, o cálculo será feito sobre todos os valores possíveis, dadas as seleções atuais, e não apenas sobre os pertinentes ao valor dimensional atual, isto é, desconsiderando as dimensões do gráfico. Usando **TOTAL [<fld {.fld}>]**, em que o qualificador **TOTAL** é seguido por uma lista de um ou mais nomes de campo como um subconjunto das variáveis de dimensão de gráfico, você cria um subconjunto dos valores possíveis totais.

Limitações:

O parâmetro da função de agregação não deve conter outras funções de agregação, a menos que essas agregações internas contenham o qualificador **TOTAL**. Para agregações aninhadas mais avançadas, use a função avançada **Aggr** junto com uma dimensão especificada.

Exemplos e resultados:

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

Exemplos de funções

Exemplo	Resultado
`FractileExc (Sales, 0.75)`	Para uma tabela incluindo a dimensão `Customer` e a medição `FractileExc([Sales])`, se **Totals** for mostrado, o resultado será 75.25. Esse é o ponto da distribuição de valores de `Sales` do qual 75% dos valores ficam abaixo.

Exemplo	Resultado
FractileExc (TOTAL Sales, 0.75))	75,25 para todos os valores de Customer, porque o qualificador TOTAL significa que as dimensões são desconsideradas.
FractileExc (DISTINCT Sales, 0.75)	73,50 para o total, porque usar o qualificador DISTINCT significa que apenas valores únicos em Sales para cada Customer são avaliados.

Dados usados nos exemplos:

```
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 '|');
```

Consulte também:

Aggr - função de gráfico (page 552)

Kurtosis

Kurtosis() retorna a contagem da expressão sobre um número de registros definidos por uma cláusula **group by**.

Sintaxe:

```
Kurtosis([distinct ] expr )
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
expr	A expressão ou campo que contém os dados a serem medidos.
distinct	Se a palavra **distinct** aparecer antes da expressão, todas as duplicatas serão ignoradas.

Exemplos e resultados:

Adicione o script de exemplo ao seu aplicativo e execute-o. Para ver o resultado, adicione os campos listados na coluna de resultados a uma pasta no seu aplicativo.

Dados resultantes

Exemplo	Resultado

`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;`
	Na tabela com as dimensões `Type`, `MyKurtosis1` e `MyKurtosis2`, os resultados dos cálculos de Kurtosis() no script de carregamento de dados são:
	
	`Type MyKurtosis1 MyKurtosis2`
	
	`Comparison -1.1612957 -1.4982366`
	
	`Observation -1.1148768 -0.93540144`

Kurtosis - função de gráfico

Kurtosis() encontra a curtose do intervalo de dados agregados na expressão ou campo repetidos nas dimensões de gráfico.

Sintaxe:

```
Kurtosis([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr)
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
expr	A expressão ou campo que contém os dados a serem medidos.
SetExpression	Por padrão, a função de agregação agregará um conjunto de registros possíveis definidos pela seleção. Um conjunto de registros alternativos pode ser definido por uma expressão de análise.
DISTINCT	Se a palavra **DISTINCT** aparecer antes dos argumentos de função, as duplicatas resultantes da avaliação dos argumentos de função serão ignoradas.
TOTAL	Se a palavra **TOTAL** ocorrer antes dos argumentos da função, o cálculo será feito sobre todos os valores possíveis, dadas as seleções atuais, e não apenas sobre os pertinentes ao valor dimensional atual, isto é, desconsiderando as dimensões do gráfico. Usando **TOTAL [<fld {.fld}>]**, em que o qualificador **TOTAL** é seguido por uma lista de um ou mais nomes de campo como um subconjunto das variáveis de dimensão de gráfico, você cria um subconjunto dos valores possíveis totais.

Limitações:

O parâmetro da função de agregação não deve conter outras funções de agregação, a menos que essas agregações internas contenham o qualificador **TOTAL**. Para agregações aninhadas mais avançadas, use a função avançada **Aggr** junto com uma dimensão especificada.

Exemplos e resultados:

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

Exemplos de funções

Exemplo	Resultado
`Kurtosis (Value)`	Para uma tabela incluindo a dimensão `Type` e a medição `Kurtosis(Value)`, se **Totals** for mostrado para a tabela e a formatação dos números for definida para 3 números significativos, o resultado será 1,252. Para `Comparison` é 1,161 e para `Observation` é 1,115.
`Kurtosis (TOTAL Value))`	1.252 para todos os valores de `Type`, porque o qualificador TOTAL significa que as dimensões são desconsideradas.

Dados usados nos exemplos:

```
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 '|');
```

Consulte também:

Avg - função de gráfico (page 404)

LINEST_B

LINEST_B() retorna o valor b (intercepção y) agregado de uma regressão linear definida pela equação y=mx+b para uma série de coordenadas representadas por números pareados em x-expression e y-expression iteradas em um número de registros, conforme definido por uma cláusula **group by**.

Sintaxe:

```
LINEST_B (y_value, x_value[, y0 [, x0 ]])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
y_value	A expressão ou campo que contém o intervalo de valores y conhecidos a ser medido.
x_value	A expressão ou campo que contém o intervalo de valores x conhecidos a ser medido.
y(0), x(0)	Um valor opcional y0 pode ser declarado, forçando a linha de regressão a passar pelo eixo y em um determinado ponto. Declarando y0 e x0, é possível forçar a linha de regressão a passar por uma única coordenada fixa. A menos que y0 e x0 sejam declarados, a função exigirá pelo menos dois pares de dados válidos para efetuar o cálculo. Se y0 e x0 forem declarados, será necessário um único par de dados.

Limitações:

Valores de texto, valores NULLe valores ausentes em qualquer uma das duas partes de um par de dados fazem com que o par de dados inteiro seja ignorado.

Consulte também:

Exemplos de como usar funções linest (page 463)

LINEST_B - função de gráfico

LINEST_B() retorna o valor agregado b (intercepção y) de uma regressão linear definida pela equação y=mx+b de uma série de coordenadas representadas por números emparelhados nas expressões dadas pelas expressões **x_value** e **y_value**, repetidas nas dimensões de gráfico.

Sintaxe:

```
LINEST_B([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value
[, y0_const [ , x0_const]])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
y_value	A expressão ou campo que contém o intervalo de valores y conhecidos a ser medido.
x_value	A expressão ou campo que contém o intervalo de valores x conhecidos a ser medido.

Argumento	Descrição
y0_const, x0_const	Um valor opcional y0 pode ser declarado, forçando a linha de regressão a passar pelo eixo y em um determinado ponto. Declarando y0 e x0, é possível forçar a linha de regressão a passar por uma única coordenada fixa. *A menos que y0 e x0 sejam declarados, a função exigirá pelo menos dois pares de dados válidos para efetuar o cálculo. Se y0 e x0 forem declarados, será necessário um único par de dados.*
SetExpression	Por padrão, a função de agregação agregará um conjunto de registros possíveis definidos pela seleção. Um conjunto de registros alternativos pode ser definido por uma expressão de análise.
DISTINCT	Se a palavra **DISTINCT** aparecer antes dos argumentos de função, as duplicatas resultantes da avaliação dos argumentos de função serão ignoradas.
TOTAL	Se a palavra **TOTAL** ocorrer antes dos argumentos da função, o cálculo será feito sobre todos os valores possíveis, dadas as seleções atuais, e não apenas sobre os pertinentes ao valor dimensional atual, isto é, desconsiderando as dimensões do gráfico. Usando **TOTAL [<fld {.fld}>]**, em que o qualificador **TOTAL** é seguido por uma lista de um ou mais nomes de campo como um subconjunto das variáveis de dimensão de gráfico, você cria um subconjunto dos valores possíveis totais.

Limitações:

O parâmetro da função de agregação não deve conter outras funções de agregação, a menos que essas agregações internas contenham o qualificador **TOTAL**. Para agregações aninhadas mais avançadas, use a função avançada **Aggr** junto com uma dimensão especificada.

Valores de texto, valores NULLe valores ausentes em qualquer uma das duas partes de um par de dados fazem com que o par de dados inteiro seja ignorado.

Consulte também:

Exemplos de como usar funções linest (page 463)
Avg - função de gráfico (page 404)

LINEST_DF

LINEST_DF() retorna os graus de liberdade agregados de uma regressão linear definida pela equação y=mx+b para uma série de coordenadas representadas por números pareados em x-expression e y-expression iteradas em um número de registros, conforme definido por uma cláusula **group by**.

Sintaxe:

```
LINEST_DF (y_value, x_value[, y0 [, x0 ]])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
y_value	A expressão ou campo que contém o intervalo de valores y conhecidos a ser medido.
x_value	A expressão ou campo que contém o intervalo de valores x conhecidos a ser medido.
y(0), x(0)	Um valor opcional y0 pode ser declarado, forçando a linha de regressão a passar pelo eixo y em um determinado ponto. Declarando y0 e x0, é possível forçar a linha de regressão a passar por uma única coordenada fixa. A menos que y0 e x0 sejam declarados, a função exigirá pelo menos dois pares de dados válidos para efetuar o cálculo. Se y0 e x0 forem declarados, será necessário um único par de dados.

Limitações:

Valores de texto, valores NULLe valores ausentes em qualquer uma das duas partes de um par de dados fazem com que o par de dados inteiro seja ignorado.

Consulte também:

Exemplos de como usar funções linest (page 463)

LINEST_DF - função de gráfico

LINEST_DF() retorna os graus de liberdade agregados de uma regressão linear definida pela equação y=mx+b de uma série de coordenadas representadas por números emparelhados nas expressões dadas por **x_value** e **y_value**, repetidos nas dimensões de gráfico.

Sintaxe:

```
LINEST_DF([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_
value [, y0_const [, x0_const]])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
y_value	A expressão ou campo que contém o intervalo de valores y conhecidos a ser medido.
x_value	A expressão ou campo que contém o intervalo de valores x conhecidos a ser medido.

Argumento	Descrição
y0, x0	Um valor opcional y0 pode ser declarado, forçando a linha de regressão a passar pelo eixo y em um determinado ponto. Declarando y0 e x0, é possível forçar a linha de regressão a passar por uma única coordenada fixa. *A menos que y0 e x0 sejam declarados, a função exigirá pelo menos dois pares de dados válidos para efetuar o cálculo. Se y0 e x0 forem declarados, será necessário um único par de dados.*
SetExpression	Por padrão, a função de agregação agregará um conjunto de registros possíveis definidos pela seleção. Um conjunto de registros alternativos pode ser definido por uma expressão de análise.
DISTINCT	Se a palavra **DISTINCT** aparecer antes dos argumentos de função, as duplicatas resultantes da avaliação dos argumentos de função serão ignoradas.
TOTAL	Se a palavra **TOTAL** ocorrer antes dos argumentos da função, o cálculo será feito sobre todos os valores possíveis, dadas as seleções atuais, e não apenas sobre os pertinentes ao valor dimensional atual, isto é, desconsiderando as dimensões do gráfico. Usando **TOTAL [<fld {.fld}>]**, em que o qualificador **TOTAL** é seguido por uma lista de um ou mais nomes de campo como um subconjunto das variáveis de dimensão de gráfico, você cria um subconjunto dos valores possíveis totais.

Limitações:

O parâmetro da função de agregação não deve conter outras funções de agregação, a menos que essas agregações internas contenham o qualificador **TOTAL**. Para agregações aninhadas mais avançadas, use a função avançada **Aggr** junto com uma dimensão especificada.

Valores de texto, valores NULLe valores ausentes em qualquer uma das duas partes de um par de dados fazem com que o par de dados inteiro seja ignorado.

Consulte também:

Exemplos de como usar funções linest (page 463)
Avg - função de gráfico (page 404)

LINEST_F

Essa função de script retorna a estatística F agregada ($r^2/(1-r^2)$) de uma regressão linear definida pela equação y=mx+b para uma série de coordenadas representadas por números pareados em x-expression e y-expression iteradas em um número de registros, conforme definido por uma cláusula **group by**.

Sintaxe:

```
LINEST_F (y_value, x_value[, y0 [, x0 ]])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
y_value	A expressão ou campo que contém o intervalo de valores y conhecidos a ser medido.
x_value	A expressão ou campo que contém o intervalo de valores x conhecidos a ser medido.
y(0), x(0)	Um valor opcional y0 pode ser declarado, forçando a linha de regressão a passar pelo eixo y em um determinado ponto. Declarando y0 e x0, é possível forçar a linha de regressão a passar por uma única coordenada fixa. A menos que y0 e x0 sejam declarados, a função exigirá pelo menos dois pares de dados válidos para efetuar o cálculo. Se y0 e x0 forem declarados, será necessário um único par de dados.

Limitações:

Valores de texto, valores NULLe valores ausentes em qualquer uma das duas partes de um par de dados fazem com que o par de dados inteiro seja ignorado.

Consulte também:

Exemplos de como usar funções linest (page 463)

LINEST_F - função de gráfico

LINEST_F() retorna a estatística F agregada (r2/(1-r2)) de uma regressão linear definida pela equação y=mx+b de uma série de coordenadas representadas por números emparelhados nas expressões dadas por **x_value** e **y_value**, repetidos nas dimensões de gráfico.

Sintaxe:

```
LINEST_F([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value
[, y0_const [, x0_const]])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
y_value	A expressão ou campo que contém o intervalo de valores y conhecidos a ser medido.
x_value	A expressão ou campo que contém o intervalo de valores x conhecidos a ser medido.

Argumento	Descrição
y0, x0	Um valor opcional y0 pode ser declarado, forçando a linha de regressão a passar pelo eixo y em um determinado ponto. Declarando y0 e x0, é possível forçar a linha de regressão a passar por uma única coordenada fixa. *A menos que y0 e x0 sejam declarados, a função exigirá pelo menos dois pares de dados válidos para efetuar o cálculo. Se y0 e x0 forem declarados, será necessário um único par de dados.*
SetExpression	Por padrão, a função de agregação agregará um conjunto de registros possíveis definidos pela seleção. Um conjunto de registros alternativos pode ser definido por uma expressão de análise.
DISTINCT	Se a palavra **DISTINCT** aparecer antes dos argumentos de função, as duplicatas resultantes da avaliação dos argumentos de função serão ignoradas.
TOTAL	Se a palavra **TOTAL** ocorrer antes dos argumentos da função, o cálculo será feito sobre todos os valores possíveis, dadas as seleções atuais, e não apenas sobre os pertinentes ao valor dimensional atual, isto é, desconsiderando as dimensões do gráfico. Usando **TOTAL [<fld {.fld}>]**, em que o qualificador **TOTAL** é seguido por uma lista de um ou mais nomes de campo como um subconjunto das variáveis de dimensão de gráfico, você cria um subconjunto dos valores possíveis totais.

Limitações:

O parâmetro da função de agregação não deve conter outras funções de agregação, a menos que essas agregações internas contenham o qualificador **TOTAL**. Para agregações aninhadas mais avançadas, use a função avançada **Aggr** junto com uma dimensão especificada.

Valores de texto, valores NULLe valores ausentes em qualquer uma das duas partes de um par de dados fazem com que o par de dados inteiro seja ignorado.

Consulte também:

Exemplos de como usar funções linest (page 463)
Avg - função de gráfico (page 404)

LINEST_M

LINEST_M() retorna o valor m (declive) agregado de uma regressão linear definida pela equação y=mx+b para uma série de coordenadas representadas por números pareados em x-expression e y-expression iteradas em um número de registros, conforme definido por uma cláusula **group by**.

Sintaxe:

```
LINEST_M (y_value, x_value[, y0 [, x0 ]])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
y_value	A expressão ou campo que contém o intervalo de valores y conhecidos a ser medido.
x_value	A expressão ou campo que contém o intervalo de valores x conhecidos a ser medido.
y(0), x(0)	Um valor opcional y0 pode ser declarado, forçando a linha de regressão a passar pelo eixo y em um determinado ponto. Declarando y0 e x0, é possível forçar a linha de regressão a passar por uma única coordenada fixa. A menos que y0 e x0 sejam declarados, a função exigirá pelo menos dois pares de dados válidos para efetuar o cálculo. Se y0 e x0 forem declarados, será necessário um único par de dados.

Limitações:

Valores de texto, valores NULLe valores ausentes em qualquer uma das duas partes de um par de dados fazem com que o par de dados inteiro seja ignorado.

Consulte também:

Exemplos de como usar funções linest (page 463)

LINEST_M - função de gráfico

LINEST_M() retorna o valor agregado m (declive) de uma regressão linear definida pela equação y=mx+b de uma série de coordenadas representadas por números emparelhados dados pelas expressões **x_value** e **y_value**, repetidos nas dimensões de gráfico.

Sintaxe:

```
LINEST_M([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value
[, y0_const [, x0_const]])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
y_value	A expressão ou campo que contém o intervalo de valores y conhecidos a ser medido.
x_value	A expressão ou campo que contém o intervalo de valores x conhecidos a ser medido.

Argumento	Descrição
y0, x0	Um valor opcional y0 pode ser declarado, forçando a linha de regressão a passar pelo eixo y em um determinado ponto. Declarando y0 e x0, é possível forçar a linha de regressão a passar por uma única coordenada fixa. *A menos que y0 e x0 sejam declarados, a função exigirá pelo menos dois pares de dados válidos para efetuar o cálculo. Se y0 e x0 forem declarados, será necessário um único par de dados.*
SetExpression	Por padrão, a função de agregação agregará um conjunto de registros possíveis definidos pela seleção. Um conjunto de registros alternativos pode ser definido por uma expressão de análise.
DISTINCT	Se a palavra **DISTINCT** aparecer antes dos argumentos de função, as duplicatas resultantes da avaliação dos argumentos de função serão ignoradas.
TOTAL	Se a palavra **TOTAL** ocorrer antes dos argumentos da função, o cálculo será feito sobre todos os valores possíveis, dadas as seleções atuais, e não apenas sobre os pertinentes ao valor dimensional atual, isto é, desconsiderando as dimensões do gráfico. Usando **TOTAL [<fld {.fld}>]**, em que o qualificador **TOTAL** é seguido por uma lista de um ou mais nomes de campo como um subconjunto das variáveis de dimensão de gráfico, você cria um subconjunto dos valores possíveis totais.

Limitações:

O parâmetro da função de agregação não deve conter outras funções de agregação, a menos que essas agregações internas contenham o qualificador **TOTAL**. Para agregações aninhadas mais avançadas, use a função avançada **Aggr** junto com uma dimensão especificada.

Valores de texto, valores NULLe valores ausentes em qualquer uma das duas partes de um par de dados fazem com que o par de dados inteiro seja ignorado.

Consulte também:

Exemplos de como usar funções linest (page 463)
Avg - função de gráfico (page 404)

LINEST_R2

LINEST_R2() retorna o valor agregado r^2 (coeficiente de determinação) de uma regressão linear definida pela equação y=mx+b para uma série de coordenadas representadas por números pareados em x-expression e y-expression iteradas em um número de registros, conforme definido por uma cláusula **group by**.

Sintaxe:

```
LINEST_R2 (y_value, x_value[, y0 [, x0 ]])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
y_value	A expressão ou campo que contém o intervalo de valores y conhecidos a ser medido.
x_value	A expressão ou campo que contém o intervalo de valores x conhecidos a ser medido.
y(0), x(0)	Um valor opcional y0 pode ser declarado, forçando a linha de regressão a passar pelo eixo y em um determinado ponto. Declarando y0 e x0, é possível forçar a linha de regressão a passar por uma única coordenada fixa. A menos que y0 e x0 sejam declarados, a função exigirá pelo menos dois pares de dados válidos para efetuar o cálculo. Se y0 e x0 forem declarados, será necessário um único par de dados.

Limitações:

Valores de texto, valores NULLe valores ausentes em qualquer uma das duas partes de um par de dados fazem com que o par de dados inteiro seja ignorado.

Consulte também:

Exemplos de como usar funções linest (page 463)

LINEST_R2 - função de gráfico

LINEST_R2() retorna o valor r2 agregado (coeficiente de determinação) de uma regressão linear definida pela equação y=mx+b de uma série de coordenadas representadas por números emparelhados dados pelas expressões **x_value** e **y_value**, repetidos nas dimensões de gráfico.

Sintaxe:

```
LINEST_R2([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_
value[, y0_const[, x0_const]])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
y_value	A expressão ou campo que contém o intervalo de valores y conhecidos a ser medido.
x_value	A expressão ou campo que contém o intervalo de valores x conhecidos a ser medido.

Argumento	Descrição
y0, x0	Um valor opcional y0 pode ser declarado, forçando a linha de regressão a passar pelo eixo y em um determinado ponto. Declarando y0 e x0, é possível forçar a linha de regressão a passar por uma única coordenada fixa. *A menos que y0 e x0 sejam declarados, a função exigirá pelo menos dois pares de dados válidos para efetuar o cálculo. Se y0 e x0 forem declarados, será necessário um único par de dados.*
SetExpression	Por padrão, a função de agregação agregará um conjunto de registros possíveis definidos pela seleção. Um conjunto de registros alternativos pode ser definido por uma expressão de análise.
DISTINCT	Se a palavra **DISTINCT** aparecer antes dos argumentos de função, as duplicatas resultantes da avaliação dos argumentos de função serão ignoradas.
TOTAL	Se a palavra **TOTAL** ocorrer antes dos argumentos da função, o cálculo será feito sobre todos os valores possíveis, dadas as seleções atuais, e não apenas sobre os pertinentes ao valor dimensional atual, isto é, desconsiderando as dimensões do gráfico. Usando **TOTAL [<fld {.fld}>]**, em que o qualificador **TOTAL** é seguido por uma lista de um ou mais nomes de campo como um subconjunto das variáveis de dimensão de gráfico, você cria um subconjunto dos valores possíveis totais.

Limitações:

O parâmetro da função de agregação não deve conter outras funções de agregação, a menos que essas agregações internas contenham o qualificador **TOTAL**. Para agregações aninhadas mais avançadas, use a função avançada **Aggr** junto com uma dimensão especificada.

Valores de texto, valores NULLe valores ausentes em qualquer uma das duas partes de um par de dados fazem com que o par de dados inteiro seja ignorado.

Consulte também:

Exemplos de como usar funções linest (page 463)
Avg - função de gráfico (page 404)

LINEST_SEB

LINEST_SEB() retorna o erro padrão agregado do valor b de uma regressão linear definida pela equação y=mx+b para uma série de coordenadas representadas por números pareados em x-expression e y-expression iteradas em um número de registros, conforme definido por uma cláusula **group by**.

Sintaxe:

```
LINEST_SEB (y_value, x_value[, y0 [, x0 ]])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
y_value	A expressão ou campo que contém o intervalo de valores y conhecidos a ser medido.
x_value	A expressão ou campo que contém o intervalo de valores x conhecidos a ser medido.
y(0), x(0)	Um valor opcional y0 pode ser declarado, forçando a linha de regressão a passar pelo eixo y em um determinado ponto. Declarando y0 e x0, é possível forçar a linha de regressão a passar por uma única coordenada fixa. A menos que y0 e x0 sejam declarados, a função exigirá pelo menos dois pares de dados válidos para efetuar o cálculo. Se y0 e x0 forem declarados, será necessário um único par de dados.

Limitações:

Valores de texto, valores NULLe valores ausentes em qualquer uma das duas partes de um par de dados fazem com que o par de dados inteiro seja ignorado.

Consulte também:

Exemplos de como usar funções linest (page 463)

LINEST_SEB - função de gráfico

LINEST_SEB() retorna o erro padrão agregado do valor b de uma regressão linear definida pela equação y=mx+b de uma série de coordenadas representadas por números emparelhados dados pelas expressões **x_value** e **y_value**, repetidos nas dimensões de gráfico.

Sintaxe:

```
LINEST_SEB([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value[, y0_const[, x0_const]])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
y_value	A expressão ou campo que contém o intervalo de valores y conhecidos a ser medido.
x_value	A expressão ou campo que contém o intervalo de valores x conhecidos a ser medido.

Argumento	Descrição
y0, x0	Um valor opcional y0 pode ser declarado, forçando a linha de regressão a passar pelo eixo y em um determinado ponto. Declarando y0 e x0, é possível forçar a linha de regressão a passar por uma única coordenada fixa. *A menos que y0 e x0 sejam declarados, a função exigirá pelo menos dois pares de dados válidos para efetuar o cálculo. Se y0 e x0 forem declarados, será necessário um único par de dados.*
SetExpression	Por padrão, a função de agregação agregará um conjunto de registros possíveis definidos pela seleção. Um conjunto de registros alternativos pode ser definido por uma expressão de análise.
DISTINCT	Se a palavra **DISTINCT** aparecer antes dos argumentos de função, as duplicatas resultantes da avaliação dos argumentos de função serão ignoradas.
TOTAL	Se a palavra **TOTAL** ocorrer antes dos argumentos da função, o cálculo será feito sobre todos os valores possíveis, dadas as seleções atuais, e não apenas sobre os pertinentes ao valor dimensional atual, isto é, desconsiderando as dimensões do gráfico. Usando **TOTAL [<fld {.fld}>]**, em que o qualificador **TOTAL** é seguido por uma lista de um ou mais nomes de campo como um subconjunto das variáveis de dimensão de gráfico, você cria um subconjunto dos valores possíveis totais.

Limitações:

O parâmetro da função de agregação não deve conter outras funções de agregação, a menos que essas agregações internas contenham o qualificador **TOTAL**. Para agregações aninhadas mais avançadas, use a função avançada **Aggr** junto com uma dimensão especificada.

Valores de texto, valores NULLe valores ausentes em qualquer uma das duas partes de um par de dados fazem com que o par de dados inteiro seja ignorado.

Consulte também:

Exemplos de como usar funções linest (page 463)
Avg - função de gráfico (page 404)

LINEST_SEM

LINEST_SEM() retorna o erro padrão agregado do valor m de uma regressão linear definida pela equação y=mx+b para uma série de coordenadas representadas por números pareados em x-expression e y-expression iteradas em um número de registros, conforme definido por uma cláusula **group by**.

Sintaxe:

```
LINEST_SEM (y_value, x_value[, y0 [, x0 ]])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumento	Descrição
y_value	A expressão ou campo que contém o intervalo de valores y conhecidos a ser medido.
x_value	A expressão ou campo que contém o intervalo de valores x conhecidos a ser medido.
y(0), x(0)	Um valor opcional y0 pode ser declarado, forçando a linha de regressão a passar pelo eixo y em um determinado ponto. Declarando y0 e x0, é possível forçar a linha de regressão a passar por uma única coordenada fixa. A menos que y0 e x0 sejam declarados, a função exigirá pelo menos dois pares de dados válidos para efetuar o cálculo. Se y0 e x0 forem declarados, será necessário um único par de dados.

Limitações:

Valores de texto, valores NULLe valores ausentes em qualquer uma das duas partes de um par de dados fazem com que o par de dados inteiro seja ignorado.

Consulte também:

Exemplos de como usar funções linest (page 463)

LINEST_SEM - função de gráfico

LINEST_SEM() retorna o erro padrão agregado do valor m de uma regressão linear definida pela equação y=mx+b de uma série de coordenadas representadas por números emparelhados dados pelas expressões **x_value** e **y_value**, repetidos nas dimensões de gráfico.

Sintaxe:

```
LINEST_SEM([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value[, y0_const[, x0_const]])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
y_value	A expressão ou campo que contém o intervalo de valores y conhecidos a ser medido.
x_value	A expressão ou campo que contém o intervalo de valores x conhecidos a ser medido.

Argumento	Descrição
y0, x0	Um valor opcional y0 pode ser declarado, forçando a linha de regressão a passar pelo eixo y em um determinado ponto. Declarando y0 e x0, é possível forçar a linha de regressão a passar por uma única coordenada fixa. *A menos que y0 e x0 sejam declarados, a função exigirá pelo menos dois pares de dados válidos para efetuar o cálculo. Se y0 e x0 forem declarados, será necessário um único par de dados.*
SetExpression	Por padrão, a função de agregação agregará um conjunto de registros possíveis definidos pela seleção. Um conjunto de registros alternativos pode ser definido por uma expressão de análise.
DISTINCT	Se a palavra **DISTINCT** aparecer antes dos argumentos de função, as duplicatas resultantes da avaliação dos argumentos de função serão ignoradas.
TOTAL	Se a palavra **TOTAL** ocorrer antes dos argumentos da função, o cálculo será feito sobre todos os valores possíveis, dadas as seleções atuais, e não apenas sobre os pertinentes ao valor dimensional atual, isto é, desconsiderando as dimensões do gráfico. Usando **TOTAL [<fld {.fld}>]**, em que o qualificador **TOTAL** é seguido por uma lista de um ou mais nomes de campo como um subconjunto das variáveis de dimensão de gráfico, você cria um subconjunto dos valores possíveis totais.

Limitações:

O parâmetro da função de agregação não deve conter outras funções de agregação, a menos que essas agregações internas contenham o qualificador **TOTAL**. Para agregações aninhadas mais avançadas, use a função avançada **Aggr** junto com uma dimensão especificada.

Valores de texto, valores NULLe valores ausentes em qualquer uma das duas partes de um par de dados fazem com que o par de dados inteiro seja ignorado.

Consulte também:

Exemplos de como usar funções linest (page 463)
Avg - função de gráfico (page 404)

LINEST_SEY

LINEST_SEY() retorna o erro padrão agregado da estimativa y de uma regressão linear definida pela equação y=mx+b para uma série de coordenadas representadas por números pareados em x-expression e y-expression iteradas em um número de registros, conforme definido por uma cláusula **group by**.

Sintaxe:

```
LINEST_SEY (y_value, x_value[, y0 [, x0 ]])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumento	Descrição
y_value	A expressão ou campo que contém o intervalo de valores y conhecidos a ser medido.
x_value	A expressão ou campo que contém o intervalo de valores x conhecidos a ser medido.
y(0), x(0)	Um valor opcional y0 pode ser declarado, forçando a linha de regressão a passar pelo eixo y em um determinado ponto. Declarando y0 e x0, é possível forçar a linha de regressão a passar por uma única coordenada fixa. A menos que y0 e x0 sejam declarados, a função exigirá pelo menos dois pares de dados válidos para efetuar o cálculo. Se y0 e x0 forem declarados, será necessário um único par de dados.

Limitações:

Valores de texto, valores NULLe valores ausentes em qualquer uma das duas partes de um par de dados fazem com que o par de dados inteiro seja ignorado.

Consulte também:

Exemplos de como usar funções linest (page 463)

LINEST_SEY - função de gráfico

LINEST_SEY() retorna o erro padrão agregado da estimativa y de uma regressão linear definida pela equação y=mx+b de uma série de coordenadas representadas por números emparelhados dados pelas expressões **x_value** e **y_value**, repetidos nas dimensões de gráfico.

Sintaxe:

```
LINEST_SEY([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value[, y0_const[, x0_const]])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
y_value	A expressão ou campo que contém o intervalo de valores y conhecidos a ser medido.
x_value	A expressão ou campo que contém o intervalo de valores x conhecidos a ser medido.

Argumento	Descrição
y0, x0	Um valor opcional y0 pode ser declarado, forçando a linha de regressão a passar pelo eixo y em um determinado ponto. Declarando y0 e x0, é possível forçar a linha de regressão a passar por uma única coordenada fixa. *A menos que y0 e x0 sejam declarados, a função exigirá pelo menos dois pares de dados válidos para efetuar o cálculo. Se y0 e x0 forem declarados, será necessário um único par de dados.*
SetExpression	Por padrão, a função de agregação agregará um conjunto de registros possíveis definidos pela seleção. Um conjunto de registros alternativos pode ser definido por uma expressão de análise.
DISTINCT	Se a palavra **DISTINCT** aparecer antes dos argumentos de função, as duplicatas resultantes da avaliação dos argumentos de função serão ignoradas.
TOTAL	Se a palavra **TOTAL** ocorrer antes dos argumentos da função, o cálculo será feito sobre todos os valores possíveis, dadas as seleções atuais, e não apenas sobre os pertinentes ao valor dimensional atual, isto é, desconsiderando as dimensões do gráfico. Usando **TOTAL [<fld {.fld}>]**, em que o qualificador **TOTAL** é seguido por uma lista de um ou mais nomes de campo como um subconjunto das variáveis de dimensão de gráfico, você cria um subconjunto dos valores possíveis totais.

Limitações:

O parâmetro da função de agregação não deve conter outras funções de agregação, a menos que essas agregações internas contenham o qualificador **TOTAL**. Para agregações aninhadas mais avançadas, use a função avançada **Aggr** junto com uma dimensão especificada.

Valores de texto, valores NULLe valores ausentes em qualquer uma das duas partes de um par de dados fazem com que o par de dados inteiro seja ignorado.

Consulte também:

Exemplos de como usar funções linest (page 463)
Avg - função de gráfico (page 404)

LINEST_SSREG

LINEST_SSREG() retorna a soma de regressão agregada dos quadrados de uma regressão linear definida pela equação y=mx+b para uma série de coordenadas representadas por números pareados em x-expression e y-expression iteradas em um número de registros, conforme definido por uma cláusula **group by**.

Sintaxe:

```
LINEST_SSREG (y_value, x_value[, y0 [, x0 ]])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
y_value	A expressão ou campo que contém o intervalo de valores y conhecidos a ser medido.
x_value	A expressão ou campo que contém o intervalo de valores x conhecidos a ser medido.
y(0), x(0)	Um valor opcional y0 pode ser declarado, forçando a linha de regressão a passar pelo eixo y em um determinado ponto. Declarando y0 e x0, é possível forçar a linha de regressão a passar por uma única coordenada fixa. A menos que y0 e x0 sejam declarados, a função exigirá pelo menos dois pares de dados válidos para efetuar o cálculo. Se y0 e x0 forem declarados, será necessário um único par de dados.

Limitações:

Valores de texto, valores NULLe valores ausentes em qualquer uma das duas partes de um par de dados fazem com que o par de dados inteiro seja ignorado.

Consulte também:

Exemplos de como usar funções linest (page 463)

LINEST_SSREG - função de gráfico

LINEST_SSREG() retorna a soma de regressão agregada de quadrados de uma regressão linear definida pela equação y=mx+b de uma série de coordenadas representadas por números emparelhados dados pelas expressões **x_value** e **y_value**, repetidos nas dimensões de gráfico.

Sintaxe:

```
LINEST_SSREG([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value[, y0_const[, x0_const]])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
y_value	A expressão ou campo que contém o intervalo de valores y conhecidos a ser medido.
x_value	A expressão ou campo que contém o intervalo de valores x conhecidos a ser medido.

Argumento	Descrição
y0, x0	Um valor opcional y0 pode ser declarado, forçando a linha de regressão a passar pelo eixo y em um determinado ponto. Declarando y0 e x0, é possível forçar a linha de regressão a passar por uma única coordenada fixa. *A menos que y0 e x0 sejam declarados, a função exigirá pelo menos dois pares de dados válidos para efetuar o cálculo. Se y0 e x0 forem declarados, será necessário um único par de dados.*
SetExpression	Por padrão, a função de agregação agregará um conjunto de registros possíveis definidos pela seleção. Um conjunto de registros alternativos pode ser definido por uma expressão de análise.
DISTINCT	Se a palavra **DISTINCT** aparecer antes dos argumentos de função, as duplicatas resultantes da avaliação dos argumentos de função serão ignoradas.
TOTAL	Se a palavra **TOTAL** ocorrer antes dos argumentos da função, o cálculo será feito sobre todos os valores possíveis, dadas as seleções atuais, e não apenas sobre os pertinentes ao valor dimensional atual, isto é, desconsiderando as dimensões do gráfico. Usando **TOTAL [<fld {.fld}>]**, em que o qualificador **TOTAL** é seguido por uma lista de um ou mais nomes de campo como um subconjunto das variáveis de dimensão de gráfico, você cria um subconjunto dos valores possíveis totais.

Limitações:

O parâmetro da função de agregação não deve conter outras funções de agregação, a menos que essas agregações internas contenham o qualificador **TOTAL**. Para agregações aninhadas mais avançadas, use a função avançada **Aggr** junto com uma dimensão especificada.

Valores de texto, valores NULLe valores ausentes em qualquer uma das duas partes de um par de dados fazem com que o par de dados inteiro seja ignorado.

Consulte também:

Exemplos de como usar funções linest (page 463)
Avg - função de gráfico (page 404)

LINEST_SSRESID

LINEST_SSRESID() retorna a soma residual agregada dos quadrados de uma regressão linear definida pela equação y=mx+b para uma série de coordenadas representadas por números pareados em x-expression e y-expression iteradas em um número de registros, conforme definido por uma cláusula **group by**.

Sintaxe:

```
LINEST_SSRESID (y_value, x_value[, y0 [, x0 ]])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
y_value	A expressão ou campo que contém o intervalo de valores y conhecidos a ser medido.
x_value	A expressão ou campo que contém o intervalo de valores x conhecidos a ser medido.
y(0), x(0)	Um valor opcional y0 pode ser declarado, forçando a linha de regressão a passar pelo eixo y em um determinado ponto. Declarando y0 e x0, é possível forçar a linha de regressão a passar por uma única coordenada fixa. A menos que y0 e x0 sejam declarados, a função exigirá pelo menos dois pares de dados válidos para efetuar o cálculo. Se y0 e x0 forem declarados, será necessário um único par de dados.

Limitações:

Valores de texto, valores NULLe valores ausentes em qualquer uma das duas partes de um par de dados fazem com que o par de dados inteiro seja ignorado.

Consulte também:

Exemplos de como usar funções linest (page 463)

LINEST_SSRESID - função de gráfico

LINEST_SSRESID() retorna a soma residual agregada dos quadrados de uma regressão linear definida pela equação y=mx+b de uma série de coordenadas representadas por números emparelhados nas expressões dadas por **x_value** e **y_value**, repetidos nas dimensões de gráfico.

Sintaxe:

```
LINEST_SSRESID([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value,
x_value[, y0_const[, x0_const]])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
y_value	A expressão ou campo que contém o intervalo de valores y conhecidos a ser medido.
x_value	A expressão ou campo que contém o intervalo de valores x conhecidos a ser medido.

Argumento	Descrição
y0, x0	Um valor opcional y0 pode ser declarado, forçando a linha de regressão a passar pelo eixo y em um determinado ponto. Declarando y0 e x0, é possível forçar a linha de regressão a passar por uma única coordenada fixa. *A menos que y0 e x0 sejam declarados, a função exigirá pelo menos dois pares de dados válidos para efetuar o cálculo. Se y0 e x0 forem declarados, será necessário um único par de dados.*
SetExpression	Por padrão, a função de agregação agregará um conjunto de registros possíveis definidos pela seleção. Um conjunto de registros alternativos pode ser definido por uma expressão de análise.
DISTINCT	Se a palavra **DISTINCT** aparecer antes dos argumentos de função, as duplicatas resultantes da avaliação dos argumentos de função serão ignoradas.
TOTAL	Se a palavra **TOTAL** ocorrer antes dos argumentos da função, o cálculo será feito sobre todos os valores possíveis, dadas as seleções atuais, e não apenas sobre os pertinentes ao valor dimensional atual, isto é, desconsiderando as dimensões do gráfico. Usando **TOTAL [<fld {.fld}>]**, em que o qualificador **TOTAL** é seguido por uma lista de um ou mais nomes de campo como um subconjunto das variáveis de dimensão de gráfico, você cria um subconjunto dos valores possíveis totais.

Um valor opcional y0 pode ser declarado, forçando a linha de regressão a passar pelo eixo y em um determinado ponto. Declarando y0 e x0, é possível forçar a linha de regressão a passar por uma única coordenada fixa.

Limitações:

O parâmetro da função de agregação não deve conter outras funções de agregação, a menos que essas agregações internas contenham o qualificador **TOTAL**. Para agregações aninhadas mais avançadas, use a função avançada **Aggr** junto com uma dimensão especificada.

Valores de texto, valores NULLe valores ausentes em qualquer uma das duas partes de um par de dados fazem com que o par de dados inteiro seja ignorado.

Consulte também:

Exemplos de como usar funções linest (page 463)
Avg - função de gráfico (page 404)

Median

Median() retorna a mediana agregada dos valores na expressão sobre um número de registros como definido por uma cláusula **group by**.

Sintaxe:

```
Median (expr)
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
expr	A expressão ou campo que contém os dados a serem medidos.

Exemplo: Expressão de script usando mediana

Exemplo - expressão de script

Script de carregamento

Carregue os seguintes dados inline e a expressão de script no editor de carregamento de dados para esse exemplo.

```
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;
```

Criar uma visualização

Crie uma visualização de tabela em uma Qlik Sense do Qlik Cloud com **Letter** e **MyMedian** como dimensões.

Resultado

Letter	MyMedian
A	3.5
B	8

Explicação

A mediana é considerada o número “do meio” quando os números foram classificados em ordem do menor para o maior. Se o conjunto de dados tiver um número par de valores, a função retornará a média dos dois valores do meio. Neste exemplo, a mediana é calculada para cada conjunto de valores de **A** e **B**, que é 3,5 e 8, respectivamente.

Median - função de gráfico

Median() retorna o valor médio do intervalo de valores agregados na expressão repetida nas dimensões de gráfico.

Sintaxe:

```
Median([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr)
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
expr	A expressão ou campo que contém os dados a serem medidos.
SetExpression	Por padrão, a função de agregação agregará um conjunto de registros possíveis definidos pela seleção. Um conjunto de registros alternativos pode ser definido por uma expressão de análise.
DISTINCT	Se a palavra **DISTINCT** aparecer antes dos argumentos de função, as duplicatas resultantes da avaliação dos argumentos de função serão ignoradas.
TOTAL	Se a palavra **TOTAL** ocorrer antes dos argumentos da função, o cálculo será feito sobre todos os valores possíveis, dadas as seleções atuais, e não apenas sobre os pertinentes ao valor dimensional atual, isto é, desconsiderando as dimensões do gráfico. Usando **TOTAL [<fld {.fld}>]**, em que o qualificador **TOTAL** é seguido por uma lista de um ou mais nomes de campo como um subconjunto das variáveis de dimensão de gráfico, você cria um subconjunto dos valores possíveis totais.

Limitações:

O parâmetro da função de agregação não deve conter outras funções de agregação, a menos que essas agregações internas contenham o qualificador **TOTAL**. Para agregações aninhadas mais avançadas, use a função avançada **Aggr** junto com uma dimensão especificada.

Exemplo: Expressão de gráfico usando mediana

Exemplo - expressão de gráfico

Script de carregamento

Carregue os seguintes dados como um carregamento inline no editor de carregamento de dados para criar o exemplo de expressão de gráfico abaixo.

```
Load RecNo() as RowNo, Letter, Number Inline
[Letter, Number
A,1
A,3
A,4
```

```
A,9
B,2
B,8
B,9];
```

Criar uma visualização

Crie uma visualização de tabela em uma pasta do Qlik Sense com **Letter** como dimensão.

Expressão de gráfico

Adicione a seguinte expressão à tabela como uma medida.

```
Median(Number)
```

Resultado

Letter	Median(Number)
Totals	4
A	3.5
B	8

Explicação

A mediana é considerada o número “do meio” quando os números foram classificados em ordem do menor para o maior. Se o conjunto de dados tiver um número par de valores, a função retornará a média dos dois valores do meio. Neste exemplo, a mediana é calculada para cada conjunto de valores de **A** e **B**, que é 3,5 e 8, respectivamente.

A mediana dos **totais** é calculada a partir de todos os valores, que é igual a 4.

Consulte também:

Avg - função de gráfico (page 404)

MutualInfo - função de gráfico

MutualInfo calcula as informações mútuas (MI) entre dois campos ou entre valores agregados em **Aggr()**.

MutualInfo retorna as informações mútuas agregadas para dois conjuntos de dados. Isso permite a análise do determinante principal entre um campo e um determinante potencial. Informações mútuas medem o relacionamento entre os conjuntos de dados e são agregadas para valores de pares (x, y) iterados nas dimensões do gráfico. Informações mútuas são medidas entre 0 e 1 e podem ser formatadas como um valor de percentil. O **MutualInfo** é definido por seleções ou por uma expressão de conjunto.

MutualInfo permite diferentes tipos de análise de MI:

- MI em pares: Calcule a MI entre um campo de determinante e um campo de meta.
- Discriminação de determinantes por valor: A MI é calculada entre valores de campo individuais nos campos de determinante e de meta.
- Seleção de recursos: Use **MutualInfo** em um gráfico de grade para criar uma matriz na qual todos os campos são comparados entre si com base na MI.

MutualInfo não indica necessariamente causalidade entre campos que compartilham informações mútuas. Dois campos podem compartilhar informações mútuas, mas podem não ser determinantes iguais um para o outro. Por exemplo, ao comparar as vendas de sorvetes e a temperatura externa, **MutualInfo** mostrará informações mútuas entre os dois. Porém, não indicará se é a temperatura externa que impulsiona as vendas de sorvetes, o que é provável, ou se são as vendas de sorvete que impulsionam a temperatura externa, o que é improvável.

Ao calcular informações mútuas, as associações afetam a correspondência entre e a frequência de valores de campos que são de tabelas diferentes.

Os valores retornados para os mesmos campos ou seleções podem variar um pouco. Isso acontece porque cada chamada **MutualInfo** opera em uma amostra selecionada aleatoriamente e devido à aleatoriedade inerente do algoritmo **MutualInfo**.

MutualInfo pode ser aplicado à função **Aggr()**.

Sintaxe:

```
MutualInfo({SetExpression}] [DISTINCT] [TOTAL] field1, field2 , datatype [,
breakdownbyvalue [, samplesize ]])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
field1, field2	As expressões ou campos que contêm os dois conjuntos de amostras para os quais as informações mútuas devem ser medidas.
datatype	Os tipos de dados contidos na meta e no determinante, 1 ou 'dd' para discretos:discretos 2 ou 'cc' para contínuos:contínuos 3 ou 'cd' para contínuos:discretos 4 ou 'dc' para discretos:contínuos Tipos de dados não diferenciam maiúsculas de minúsculas.

Argumento	Descrição
breakdownbyvalue	Um valor estático correspondente a um valor no determinante. Se fornecido, o cálculo será aplicado à contribuição de MI para esse valor. Você pode usar **ValueList()** ou **ValueLoop()**. Se **Null()** for adicionado, o cálculo será aplicado à MI em geral para todos os valores no determinante. A divisão por valor requer que o determinante contenha dados discretos.
samplesize	O número de valores para obtenção de amostras na meta e no determinante. A amostragem é aleatória. **MutualInfo** requer um tamanho mínimo de amostra de 80. Por padrão, **MutualInfo** apenas define amostras de até 10.000 pares de dados, já que **MutualInfo** pode consumir muitos recursos. Você pode especificar um número maior de pares de dados no tamanho da amostra. Se **MutualInfo** atingir o tempo limite, reduza o tamanho da amostra.
SetExpression	Por padrão, a função de agregação agregará um conjunto de registros possíveis definidos pela seleção. Um conjunto de registros alternativos pode ser definido por uma expressão de análise.
DISTINCT	Se a palavra **DISTINCT** aparecer antes dos argumentos de função, as duplicatas resultantes da avaliação dos argumentos de função serão ignoradas.
TOTAL	Se a palavra **TOTAL** ocorrer antes dos argumentos da função, o cálculo será feito sobre todos os valores possíveis, dadas as seleções atuais, e não apenas sobre os pertinentes ao valor dimensional atual, isto é, desconsiderando as dimensões do gráfico. Usando **TOTAL [<fld {.fld}>]**, em que o qualificador **TOTAL** é seguido por uma lista de um ou mais nomes de campo como um subconjunto das variáveis de dimensão de gráfico, você cria um subconjunto dos valores possíveis totais.

Limitações:

Valores de texto, valores NULLe valores ausentes em qualquer uma das duas partes de um par de dados fazem com que o par de dados inteiro seja ignorado.

Exemplos e resultados:

Adicione o script de exemplo ao seu aplicativo e execute-o. Para ver o resultado, adicione os campos listados na coluna de resultados a uma pasta no seu aplicativo.

Exemplos de funções

Exemplo	Resultado
`mutualinfo (Age, Salary, 1)`	Para uma tabela incluindo a dimensão `Employee name` e a medição `mutualinfo(Age, Salary, 1)`, o resultado é 0,99820986. O resultado é exibido apenas para a célula de totais.

Exemplo	Resultado
`mutualinfo (TOTAL Age, Salary, 1, null(), 81)`	Se você criar um painel de filtro com a dimensão Gender e fizer seleções nele, verá o resultado 0,99805677 quando Female for selecionado e 0,99847373 se Male for selecionado. Isso ocorre porque a seleção exclui todos os resultados que não pertencem a outro valor de Gender.
`mutualinfo (TOTAL Age, Gender, 1, ValueLoop (25,35))`	0.68196996. Selecionar qualquer valor de Gender mudará isso para 0.
`mutualinfo({1} TOTAL Age, Salary, 1, null())`	0.99820986. Isso é independente das seleções. A expressão de conjunto {1} ignora todas as seleções e dimensões.

Dados usados nos exemplos:

```
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() retorna a assimetria da expressão sobre um número de registro definido por uma cláusula **group by**.

Sintaxe:

```
Skew([ distinct] expr)
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
expr	A expressão ou campo que contém os dados a serem medidos.
DISTINCT	Se a palavra **distinct** aparecer antes da expressão, todas as duplicatas serão ignoradas.

Exemplos e resultados:

Adicione o script de exemplo ao seu aplicativo e execute-o. Em seguida, crie uma tabela estática com `Type` e *MySkew* como dimensões.

Dados resultantes

Exemplo	Resultado

`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;`
	Os resultados do cálculo Skew() são:
	• `Type` é `MySkew`
	• `Comparison` é 0.86414768
	• `Observation` é 0.32625351

Skew - função de gráfico

Skew() retorna a assimetria agregada da expressão ou campo repetida nas dimensões de gráfico.

Sintaxe:

```
Skew([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr)
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
expr	A expressão ou campo que contém os dados a serem medidos.
SetExpression	Por padrão, a função de agregação agregará um conjunto de registros possíveis definidos pela seleção. Um conjunto de registros alternativos pode ser definido por uma expressão de análise.
DISTINCT	Se a palavra **DISTINCT** aparecer antes dos argumentos de função, as duplicatas resultantes da avaliação dos argumentos de função serão ignoradas.
TOTAL	Se a palavra **TOTAL** ocorrer antes dos argumentos da função, o cálculo será feito sobre todos os valores possíveis, dadas as seleções atuais, e não apenas sobre os pertinentes ao valor dimensional atual, isto é, desconsiderando as dimensões do gráfico. Usando **TOTAL [<fld {.fld}>]**, em que o qualificador **TOTAL** é seguido por uma lista de um ou mais nomes de campo como um subconjunto das variáveis de dimensão de gráfico, você cria um subconjunto dos valores possíveis totais.

Limitações:

O parâmetro da função de agregação não deve conter outras funções de agregação, a menos que essas agregações internas contenham o qualificador **TOTAL**. Para agregações aninhadas mais avançadas, use a função avançada **Aggr** junto com uma dimensão especificada.

Exemplos e resultados:

Adicione o script de exemplo ao seu aplicativo e execute-o. Em seguida, crie uma tabela estática com `Type` como dimensão e `Skew(Value)` como medida.

`Totals` deve ser ativado nas propriedades da tabela.

Exemplo	Resultado
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 '|');	Os resultados do cálculo Skew(Value) são: • Totalé 0.23522195 • Comparison é 0.86414768 • Observation é 0.32625351

Consulte também:

Avg - função de gráfico (page 404)

Stdev

Stdev() retorna o desvio padrão dos valores dados pela expressão sobre um número de registro conforme definido por uma cláusula **group by**.

Sintaxe:

```
Stdev([distinct] expr)
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
expr	A expressão ou campo que contém os dados a serem medidos.
distinct	Se a palavra **distinct** aparecer antes da expressão, todas as duplicatas serão ignoradas.

Exemplos e resultados:

Adicione o script de exemplo ao seu aplicativo e execute-o. Em seguida, crie uma tabela estática com `Type` e `MyStdev` como dimensões.

Dados resultantes

Exemplo	Resultado

`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;`
	Os resultados do cálculo Stdev() são:
	• `Type` é `MyStdev`
	• `Comparison` é 14.61245
	• `Observation` é 12.507997

Stdev - função de gráfico

Stdev() encontra o desvio padrão do intervalo de dados agregados na expressão ou campo repetidos nas dimensões de gráfico.

Sintaxe:

```
Stdev([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr)
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
expr	A expressão ou campo que contém os dados a serem medidos.
SetExpression	Por padrão, a função de agregação agregará um conjunto de registros possíveis definidos pela seleção. Um conjunto de registros alternativos pode ser definido por uma expressão de análise.
DISTINCT	Se a palavra **DISTINCT** aparecer antes dos argumentos de função, as duplicatas resultantes da avaliação dos argumentos de função serão ignoradas.
TOTAL	Se a palavra **TOTAL** ocorrer antes dos argumentos da função, o cálculo será feito sobre todos os valores possíveis, dadas as seleções atuais, e não apenas sobre os pertinentes ao valor dimensional atual, isto é, desconsiderando as dimensões do gráfico. Usando **TOTAL [<fld {.fld}>]**, em que o qualificador **TOTAL** é seguido por uma lista de um ou mais nomes de campo como um subconjunto das variáveis de dimensão de gráfico, você cria um subconjunto dos valores possíveis totais.

Limitações:

O parâmetro da função de agregação não deve conter outras funções de agregação, a menos que essas agregações internas contenham o qualificador **TOTAL**. Para agregações aninhadas mais avançadas, use a função avançada **Aggr** junto com uma dimensão especificada.

Exemplos e resultados:

Adicione o script de exemplo ao seu aplicativo e execute-o. Em seguida, crie uma tabela estática com `Type` como dimensão e `Stdev(Value)` como medida.

`Totals` deve ser ativado nas propriedades da tabela.

Exemplo	Resultado
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 '|');	Os resultados do cálculo Stdev(Value) são: • Totalé 15.47529 • Comparison é 14.61245 • Observation é 12.507997

Consulte também:

- *Avg - função de gráfico (page 404)*
- *STEYX - função de gráfico (page 461)*

Sterr

Sterr() retorna o erro padrão agregado (stdev/sqrt(n)) de uma série de valores representados pela expressão iterada sobre um número de registros, conforme definido por uma cláusula **group by**.

Sintaxe:

```
Sterr ([distinct] expr)
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
expr	A expressão ou campo que contém os dados a serem medidos.
distinct	Se a palavra **distinct** aparecer antes da expressão, todas as duplicatas serão ignoradas.

Limitações:

Os valores de texto, os valores NULL e os valores ausentes são ignorados.

Exemplos e resultados:

Adicione o script de exemplo ao seu aplicativo e execute-o. Para ver o resultado, adicione os campos listados na coluna de resultados a uma pasta no seu aplicativo.

Dados resultantes

Exemplo	Resultado

`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;`
	Na tabela com as dimensões `Type` e `MySterr`, os resultados do cálculo de Sterr() no script de carregamento de dados são:
	
	`Type MySterr`
	
	`Comparison 3.2674431`
	
	`Observation 2.7968733`

Sterr - função de gráfico

Sterr() encontra o valor do erro padrão da média, (stdev/sqrt(n)), da série de valores agregados na expressão repetida nas dimensões de gráfico.

Sintaxe:

```
Sterr([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr)
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
expr	A expressão ou campo que contém os dados a serem medidos.
SetExpression	Por padrão, a função de agregação agregará um conjunto de registros possíveis definidos pela seleção. Um conjunto de registros alternativos pode ser definido por uma expressão de análise.
DISTINCT	Se a palavra **DISTINCT** aparecer antes dos argumentos de função, as duplicatas resultantes da avaliação dos argumentos de função serão ignoradas.
TOTAL	Se a palavra **TOTAL** ocorrer antes dos argumentos da função, o cálculo será feito sobre todos os valores possíveis, dadas as seleções atuais, e não apenas sobre os pertinentes ao valor dimensional atual, isto é, desconsiderando as dimensões do gráfico. Usando **TOTAL [<fld {.fld}>]**, em que o qualificador **TOTAL** é seguido por uma lista de um ou mais nomes de campo como um subconjunto das variáveis de dimensão de gráfico, você cria um subconjunto dos valores possíveis totais.

Limitações:

O parâmetro da função de agregação não deve conter outras funções de agregação, a menos que essas agregações internas contenham o qualificador **TOTAL**. Para agregações aninhadas mais avançadas, use a função avançada **Aggr** junto com uma dimensão especificada.

Os valores de texto, os valores NULL e os valores ausentes são ignorados.

Exemplos e resultados:

Adicione o script de exemplo ao seu aplicativo e execute-o. Em seguida, crie uma tabela estática com `Type` como dimensão e `Sterr(Value)` como medida.

`Totals` deve ser ativado nas propriedades da tabela.

Exemplo	Resultado
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 '|');	Os resultados do cálculo Sterr(Value) são: • Totalé 2.4468583 • Comparison é 3.2674431 • Observation é 2.7968733

Consulte também:

- *Avg - função de gráfico (page 404)*
- *STEYX - função de gráfico (page 461)*

STEYX

STEYX() retorna o erro padrão agregado do valor y previsto para cada valor x na regressão de uma série de coordenadas representada por números pareados em x-expression e y-expression , com iterações em vários registros definidos por uma cláusula **group by**.

Sintaxe:

```
STEYX (y_value, x_value)
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
y_value	A expressão ou campo que contém o intervalo de valores y conhecidos a ser medido.
x_value	A expressão ou campo que contém o intervalo de valores x conhecidos a ser medido.

Limitações:

Valores de texto, valores NULLe valores ausentes em qualquer uma das duas partes de um par de dados fazem com que o par de dados inteiro seja ignorado.

Exemplos e resultados:

Adicione o script de exemplo ao seu aplicativo e execute-o. Para ver o resultado, adicione os campos listados na coluna de resultados a uma pasta no seu aplicativo.

Dados resultantes

Exemplo	Resultado
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;	Na tabela com as dimensões MySTEYX, os resultados do cálculo de STEYX() no script de carregamento de dados é 2.0714764.

STEYX - função de gráfico

STEYX() retorna o erro padrão agregado ao prever os valores y de cada valor x em uma regressão linear dada por uma série de coordenadas representadas por números emparelhados nas expressões dadas pelos **y_value** e **x_value**.

Sintaxe:

```
STEYX([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value)
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
y_value	A expressão ou campo que contém o intervalo de valores y conhecidos a ser medido.
x_value	A expressão ou campo que contém o intervalo de valores xconhecidos a ser medido.
SetExpression	Por padrão, a função de agregação agregará um conjunto de registros possíveis definidos pela seleção. Um conjunto de registros alternativos pode ser definido por uma expressão de análise.
DISTINCT	Se a palavra **DISTINCT** aparecer antes dos argumentos de função, as duplicatas resultantes da avaliação dos argumentos de função serão ignoradas.
TOTAL	Se a palavra **TOTAL** ocorrer antes dos argumentos da função, o cálculo será feito sobre todos os valores possíveis, dadas as seleções atuais, e não apenas sobre os pertinentes ao valor dimensional atual, isto é, desconsiderando as dimensões do gráfico. Usando **TOTAL [<fld {.fld}>]**, em que o qualificador **TOTAL** é seguido por uma lista de um ou mais nomes de campo como um subconjunto das variáveis de dimensão de gráfico, você cria um subconjunto dos valores possíveis totais.

Limitações:

O parâmetro da função de agregação não deve conter outras funções de agregação, a menos que essas agregações internas contenham o qualificador **TOTAL**. Para agregações aninhadas mais avançadas, use a função avançada **Aggr** junto com uma dimensão especificada.

Valores de texto, valores NULLe valores ausentes em qualquer uma das duas partes de um par de dados fazem com que o par de dados inteiro seja ignorado.

Exemplos e resultados:

Adicione o script de exemplo ao seu aplicativo e execute-o. Em seguida, crie uma tabela estática com `KnownY` e `KnownX` como dimensão e `Steyx(KnownY,KnownX)` como medida

`Totals` deve ser ativado nas propriedades da tabela.

Exemplo	Resultado
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 '|');	O resultado do cálculo STEYX(KnownY,KnownX) será 2,071 (se a formatação do número for definida com 3 casas decimais).

Consulte também:

- *Avg - função de gráfico (page 404)*
- *Sterr - função de gráfico (page 457)*

Exemplos de como usar funções linest

As funções linest são usadas para encontrar os valores associados com a análise de regressão linear. Esta seção descreve como criar visualizações usando dados de amostra para encontrar os valores das funções linest disponíveis no Qlik Sense. As funções linest podem ser usadas no script de carregamento de dados e em expressões de gráfico.

Consulte os tópicos individuais de função do gráfico e do script linest para ver as descrições de sintaxe e os argumentos.

Expressões de dados e scripts usados nos exemplos

Carregue os seguintes dados inline e expressões de script no editor de carregamento de dados para os exemplos de linest() abaixo.

```
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;
```

Exemplo 1: Expressões de script usando linest

Exemplo: Expressões de script

Crie uma visualização a partir dos cálculos do script de carregamento de dados

Crie uma visualização de tabela em uma pasta do Qlik Sense com os seguintes campos como colunas:

- Linest_B
- Linest_DF
- Linest_F

- Linest_M
- Linest_R2
- Linest_SEB
- Linest_SEM
- Linest_SEY
- Linest_SSREG
- Linest_SSRESID

Resultado

A tabela contendo resultados dos cálculos de linest feitos no script de carregamento de dados devem ficar assim:

Tabela de resultados

Linest_B	Linest_DF	Linest_F	Linest_M	Linest_R2	Linest_SEB
-35.047	18	20.788	8.605	0.536	22.607

Tabela de resultados

Linest_SEM	Linest_SEY	Linest_SSREG	Linest_SSRESID
1.887	48.666	49235.014	42631.186

Exemplo 2: Expressões de gráfico usando linest

Exemplo: Expressões de gráfico

Crie uma visualização de tabela em uma pasta do Qlik Sense com os seguintes campos como dimensões:

```
ValueList('Linest_b', 'Linest_df','Linest_f', 'Linest_m','Linest_r2','Linest_SEB','Linest_
SEM','Linest_SEY','Linest_SSREG','Linest_SSRESID')
```

Essa expressão usa a função de dimensões sintéticas para criar rótulos para as dimensões com os nomes das funções linest. Você pode alterar o rótulo para **Linest functions** para economizar espaço.

Adicione a seguinte expressão à tabela como uma medida:

```
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) )
```

Essa expressão exibe o valor do resultado de cada função linest em relação ao nome correspondente na dimensão sintética. O resultado de `Linest_b(Y,X)` é exibido ao lado de **linest_b** e assim por diante.

Resultado

Tabela de resultados

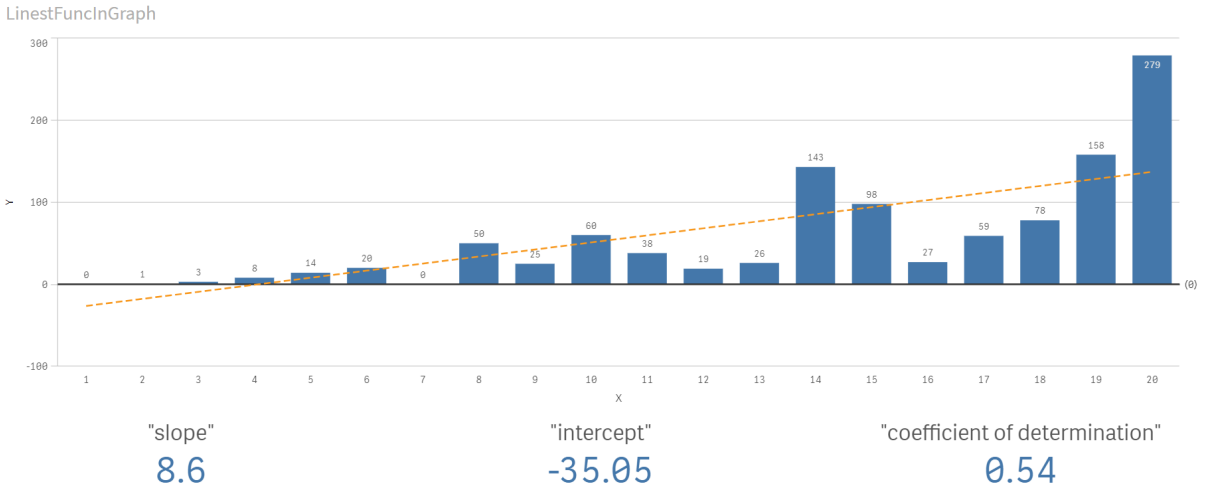
Linest functions	Linest function results
Linest_b	-35.047
Linest_df	18
Linest_f	20.788
Linest_m	8.605
Linest_r2	0.536
Linest_SEB	22.607
Linest_SEM	1.887
Linest_SEY	48.666
Linest_SSREG	49235.014
Linest_SSRESID	42631.186

Exemplo 3: Expressões de gráfico usando linest

Exemplo: Expressões de gráfico

1. Crie uma visualização de gráfico de barras em uma pasta Qlik Sense com **X** como dimensão e **Y** como medida.
2. Adicione uma linha de tendência linear à medida Y.
3. Adicione uma visualização de KPI à pasta.
    1. Adicione *inclinação* como um rótulo para o KPI.
    2. Adicione `sum(Linest_M)` como uma expressão para o KPI.
4. Adicione uma segunda visualização de KPI à pasta.
    1. Adicione *interceptação* como um rótulo para o KPI.
    2. Adicione `Sum(Linest_B)` como uma expressão para o KPI.
5. Adicione uma terceira visualização de KPI à pasta.
    1. Adicione *coeficiente de determinação* como um rótulo para o KPI.
    2. Adicione `Sum(Linest_R2)` como uma expressão para o KPI.

Resultado



Explicação

O gráfico de barras mostra a plotagem dos dados de X e Y. As funções linest() relevantes fornecem valores para a equação de regressão linear na qual a linha de tendência se baseia, ou seja, y = m * x + b. A equação usa o método de "quadrados mínimos" para calcular uma linha reta (linha de tendência) retornando uma matriz que descreve uma linha que melhor se ajusta aos dados.

Os KPIs exibem os resultados das funções linest() **sum(Linest_M)** para inclinação e **sum(Linest_M)** para a interceptação Y, que são variáveis na equação de regressão linear, e o valor R2 agregado correspondente para o coeficiente de determinação.

Funções estatísticas de teste

Funções estatísticas de teste podem ser usadas no script de carregamento de dados e nas expressões de gráfico, embora a sintaxe seja diferente.

Funções de teste Chi-2

Geralmente usadas no estudo de variáveis qualitativas. Pode-se comparar as frequências observadas em uma tabela de frequências unidirecionais com as frequências esperadas ou estudar a relação entre duas variáveis em uma tabela de contingência.

Funções de teste T

Funções de teste T são usadas para exame estatístico de duas médias populacionais. Um teste t de duas amostras examina se duas amostras são diferentes. Normalmente é usado quando duas distribuições normais têm variâncias desconhecidas e quando um experimento usa um tamanho pequeno de amostra.

Funções de teste Z

Um exame estatístico de duas médias populacionais. Um teste z de duas amostras examina se duas amostras são diferentes. Normalmente, é usado quando duas distribuições normais têm variâncias conhecidas e quando um experimento usa um tamanho grande de amostra.

Funções de teste de Qui2

Geralmente usadas no estudo de variáveis qualitativas. Pode-se comparar as frequências observadas em uma tabela de frequências unidirecionais com as frequências esperadas ou estudar a relação entre duas variáveis em uma tabela de contingência.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.

Se a função for usada no script de carga dos dados, os valores serão iterados com base em um número de registros que são definidos por uma cláusula group by.

Se a função for usada para expressões de gráficos, os valores serão iterados com base nas dimensões do gráfico.

Chi2Test_chi2

Chi2Test_chi2() retorna o valor agregado do teste Qui^2 referente a uma ou duas séries de valores.

```
Chi2Test_chi2() retorna o valor agregado do teste Qui2 referente a uma ou duas séries de valores.(col, row, actual_value[, expected_value])
```

Chi2Test_df

Chi2Test_df() retorna o valor df (graus de liberdade) agregado do teste Qui^2, referente a uma ou duas séries de valores.

```
Chi2Test_df() retorna o valor df (graus de liberdade) agregado do teste Qui2, referente a uma ou duas séries de valores.(col, row, actual_value[, expected_value])
```

Chi2Test_p

Chi2Test_p() retorna o valor (significância) agregado do teste chi^2, referente a uma ou duas séries de valores.

```
Chi2Test_p - função de gráfico(col, row, actual_value[, expected_value])
```

Consulte também:

- *Funções de teste T (page 471)*
- *Funções de teste Z (page 508)*

Chi2Test_chi2

Chi2Test_chi2() retorna o valor agregado do teste Qui^2 referente a uma ou duas séries de valores.

Se a função for usada no script de carga dos dados, os valores serão iterados com base em um número de registros que são definidos por uma cláusula group by.

Se a função for usada para expressões de gráficos, os valores serão iterados com base nas dimensões do gráfico.

> *Todos os campos do sistema de Qlik Sense Todas as funções de teste chi² têm os mesmos argumentos.*

Sintaxe:

```
Chi2Test_chi2(col, row, actual_value[, expected_value])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
col, row	A coluna e linha especificada na matriz de valores que está sendo testada.
actual_value	O valor observado dos dados em **col** e **row** especificados.
expected_value	O valor esperado para a distribuição em **col** e **row** especificados.

Limitações:

Valores de texto, valores NULL e valores ausentes nas expressões de valores farão com que a função retorne um resultado NULL.

Exemplos:

```
Chi2Test_chi2( Grp, Grade, Count )
Chi2Test_chi2( Gender, Description, Observed, Expected )
```

Consulte também:

Exemplos de como usar funções chi2-test em gráficos (page 524)
Exemplos de como usar funções chi2-test no script de carregamento de dados (page 528)

Chi2Test_df

Chi2Test_df() retorna o valor df (graus de liberdade) agregado do teste Qui², referente a uma ou duas séries de valores.

Se a função for usada no script de carga dos dados, os valores serão iterados com base em um número de registros que são definidos por uma cláusula group by.

Se a função for usada para expressões de gráficos, os valores serão iterados com base nas dimensões do gráfico.

Todos os campos do sistema de Qlik Sense Todas as funções de teste chi² têm os mesmos argumentos.

Sintaxe:

```
Chi2Test_df(col, row, actual_value[, expected_value])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
col, row	A coluna e linha especificada na matriz de valores que está sendo testada.
actual_value	O valor observado dos dados em **col** e **row** especificados.
expected_value	O valor esperado para a distribuição em **col** e **row** especificados.

Limitações:

Valores de texto, valores NULL e valores ausentes nas expressões de valores farão com que a função retorne um resultado NULL.

Exemplos:

```
Chi2Test_df( Grp, Grade, Count )
Chi2Test_df( Gender, Description, Observed, Expected )
```

Consulte também:

Exemplos de como usar funções chi2-test em gráficos (page 524)
Exemplos de como usar funções chi2-test no script de carregamento de dados (page 528)

Chi2Test_p - função de gráfico

Chi2Test_p() retorna o valor (significância) agregado do teste chi², referente a uma ou duas séries de valores. O teste pode ser feito sobre os valores em **actual_value**, verificando se existem variações na matriz especificada de **col** e **row**, ou comparando os valores em **actual_value** com os valores correspondentes em **expected_value**, se especificados.

Se a função for usada no script de carga dos dados, os valores serão iterados com base em um número de registros que são definidos por uma cláusula group by.

Se a função for usada para expressões de gráficos, os valores serão iterados com base nas dimensões do gráfico.

Todos os campos do sistema de Qlik Sense Todas as funções de teste chi^2 têm os mesmos argumentos.

Sintaxe:

```
Chi2Test_p(col, row, actual_value[, expected_value])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
col, row	A coluna e linha especificada na matriz de valores que está sendo testada.
actual_value	O valor observado dos dados em **col** e **row** especificados.
expected_value	O valor esperado para a distribuição em **col** e **row** especificados.

Limitações:

Valores de texto, valores NULL e valores ausentes nas expressões de valores farão com que a função retorne um resultado NULL.

Exemplos:

```
Chi2Test_p( Grp, Grade, Count )
Chi2Test_p( Gender, Description, Observed, Expected )
```

Consulte também:

Exemplos de como usar funções chi2-test em gráficos (page 524)
Exemplos de como usar funções chi2-test no script de carregamento de dados (page 528)

Funções de teste T

Funções de teste T são usadas para exame estatístico de duas médias populacionais. Um teste t de duas amostras examina se duas amostras são diferentes. Normalmente é usado quando duas distribuições normais têm variâncias desconhecidas e quando um experimento usa um tamanho pequeno de amostra.

Nas seções a seguir, as funções do teste estatístico teste T são agrupadas de acordo com o teste de Student de amostra que se aplica a cada tipo de função.

Criando um relatório típico de t-test (page 530)

Dois testes T de amostras independentes

As funções a seguir aplicam-se a testes T de Student de duas amostras independentes.

ttest_conf

TTest_conf retorna o valor agregado do intervalo de confiança do teste T para duas amostras independentes.

```
TTest_conf retorna o valor agregado do intervalo de confiança do teste T para duas amostras independentes. ( grp, value [, sig[, eq_var]])
```

ttest_df

TTest_df() retorna o valor (graus de liberdade) agregado do teste T de estudante referente a duas séries independentes de valores.

```
TTest_df() retorna o valor (graus de liberdade) agregado do teste T de estudante referente a duas séries independentes de valores. (grp, value [, eq_var)
```

ttest_dif

TTest_dif() é uma função numérica que retorna a diferença média agregada do teste T de estudantes, referente a duas séries independentes de valores.

```
TTest_dif() é uma função numérica que retorna a diferença média agregada do teste T de estudantes, referente a duas séries independentes de valores. (grp, value)
```

ttest_lower

TTest_lower() retorna o valor agregado da extremidade inferior do intervalo de confiança, referente a duas séries independentes de valores.

```
TTest_lower() retorna o valor agregado da extremidade inferior do intervalo de confiança, referente a duas séries independentes de valores. (grp, value [, sig[, eq_var]])
```

ttest_sig

TTest_sig() retorna o nível de significância agregado do teste T bicaudal de estudantes, referente a duas séries independentes de valores.

```
TTest_sig() retorna o nível de significância agregado do teste T bicaudal de estudantes, referente a duas séries independentes de valores. (grp, value [, eq_var])
```

ttest_sterr

TTest_sterr() retorna o erro padrão agregado da diferença média do teste T de estudantes, referente a duas séries independentes de valores.

```
TTest_sterr() retorna o erro padrão agregado da diferença média do teste T de estudantes, referente a duas séries independentes de valores. (grp, value [, eq_var])
```

ttest_t

TTest_t() retorna o valor t agregado referente a duas séries independentes de valores.

TTest_t() retorna o valor t agregado referente a duas séries independentes de valores. (grp, value [, eq_var]**)**

ttest_upper

TTest_upper() retorna o valor agregado da extremidade superior do intervalo de confiança, referente a duas séries independentes de valores.

TTest_upper() retorna o valor agregado da extremidade superior do intervalo de confiança, referente a duas séries independentes de valores. (grp, value [, sig [, eq_var]]**)**

Dois testes T de amostras ponderadas independentes

As funções a seguir aplicam-se a testes T de Student de duas amostras independentes, em que a série de dados de entrada é fornecida em um formato de duas colunas ponderadas.

ttestw_conf

TTestw_conf() retorna o valor t agregado referente a duas séries independentes de valores.

TTestw_conf() retorna o valor t agregado referente a duas séries independentes de valores. (weight, grp, value [, sig[, eq_var]]**)**

ttestw_df

TTestw_df() retorna o valor df (graus de liberdade) agregado do teste T de estudante referente a duas séries independentes de valores.

TTestw_df() retorna o valor df (graus de liberdade) agregado do teste T de estudante referente a duas séries independentes de valores. (weight, grp, value [, eq_var]**)**

ttestw_dif

TTestw_dif() retorna a diferença média agregada do teste T de estudantes, referente a duas séries independentes de valores.

TTestw_dif() retorna a diferença média agregada do teste T de estudantes, referente a duas séries independentes de valores. (weight, grp, value**)**

ttestw_lower

TTestw_lower() retorna o valor agregado da extremidade inferior do intervalo de confiança, referente a duas séries independentes de valores.

TTestw_lower() retorna o valor agregado da extremidade inferior do intervalo de confiança, referente a duas séries independentes de valores. (weight, grp, value [, sig[, eq_var]]**)**

ttestw_sig

TTestw_sig() retorna o nível de significância agregado do teste T bicaudal de estudantes, referente a duas séries independentes de valores.

TTestw_sig() retorna o nível de significância agregado do teste T bicaudal de estudantes, referente a duas séries independentes de valores. (weight, grp, value [, eq_var]**)**

ttestw_sterr

TTestw_sterr() retorna o erro padrão agregado da diferença média do teste T de estudantes, referente a duas séries independentes de valores.

TTestw_sterr() retorna o erro padrão agregado da diferença média do teste T de estudantes, referente a duas séries independentes de valores. (weight, grp, value [, eq_var]**)**

ttestw_t

TTestw_t() retorna o valor t agregado referente a duas séries independentes de valores.

TTestw_t() retorna o valor t agregado referente a duas séries independentes de valores. (weight, grp, value [, eq_var]**)**

ttestw_upper

TTestw_upper() retorna o valor agregado da extremidade superior do intervalo de confiança, referente a duas séries independentes de valores.

TTestw_upper() retorna o valor agregado da extremidade superior do intervalo de confiança, referente a duas séries independentes de valores. (weight, grp, value [, sig [, eq_var]]**)**

Testes T de uma amostra

As funções a seguir aplicam-se a testes T de Student de uma amostra.

ttest1_conf

TTest1_conf() retorna o valor de intervalo de confiança agregado referente a uma série de valores.

TTest1_conf() retorna o valor de intervalo de confiança agregado referente a uma série de valores. (value [, sig]**)**

ttest1_df

TTest1_df() retorna o valor df (graus de liberdade) agregado do teste T de estudante referente a uma série de valores.

TTest1_df() retorna o valor df (graus de liberdade) agregado do teste T de estudante referente a uma série de valores. (value**)**

ttest1_dif

TTest1_dif() retorna a diferença média agregada do teste T de estudantes, referente a uma série de valores.

TTest1_dif() retorna a diferença média agregada do teste T de estudantes, referente a uma série de valores. (value**)**

ttest1_lower

TTest1_lower() retorna o valor agregado da extremidade inferior do intervalo de confiança, referente a uma série de valores.

```
TTest1_lower() retorna o valor agregado da extremidade inferior do intervalo de confiança, referente a uma série de valores. (value [, sig])
```

ttest1_sig

TTest1_sig() retorna o nível de significância agregado do teste T bicaudal de estudantes, referente a uma série de valores.

```
TTest1_sig() retorna o nível de significância agregado do teste T bicaudal de estudantes, referente a uma série de valores. (value)
```

ttest1_sterr

TTest1_sterr() retorna o erro padrão agregado da diferença média do teste T de estudantes, referente a uma série de valores.

```
TTest1_sterr() retorna o erro padrão agregado da diferença média do teste T de estudantes, referente a uma série de valores. (value)
```

ttest1_t

TTest1_t() retorna o valor t agregado para um série de valores.

```
TTest1_t() retorna o valor t agregado para um série de valores. (value)
```

ttest1_upper

TTest1_upper() retorna o valor agregado da extremidade superior do intervalo de confiança, referente a uma série de valores.

```
TTest1_upper() retorna o valor agregado da extremidade superior do intervalo de confiança, referente a uma série de valores. (value [, sig])
```

Testes T de uma amostra ponderada

As funções a seguir aplicam-se a testes T de Student de uma amostra, em que a série de dados de entrada é fornecida em um formato de duas colunas ponderadas.

ttest1w_conf

TTest1w_conf() é uma função **numérica** que retorna o valor de intervalo de confiança agregado referente a uma série de valores.

```
TTest1w_conf() é uma função numérica que retorna o valor de intervalo de confiança agregado referente a uma série de valores. (weight, value [, sig])
```

ttest1w_df

TTest1w_df() retorna o valor df (graus de liberdade) agregado do teste T de estudante referente a uma série de valores.

```
TTest1w_df() retorna o valor df (graus de liberdade) agregado do teste T de estudante referente a uma série de valores. (weight, value)
```

ttest1w_dif

TTest1w_dif() retorna a diferença média agregada do teste T de estudantes, referente a uma série de valores.

```
TTest1w_dif() retorna a diferença média agregada do teste T de estudantes,
referente a uma série de valores. (weight, value)
```

ttest1w_lower

TTest1w_lower() retorna o valor agregado da extremidade inferior do intervalo de confiança, referente a uma série de valores.

```
TTest1w_lower() retorna o valor agregado da extremidade inferior do intervalo
de confiança, referente a uma série de valores. (weight, value [, sig])
```

ttest1w_sig

TTest1w_sig() retorna o nível de significância agregado do teste T bicaudal de estudantes, referente a uma série de valores.

```
TTest1w_sig() retorna o nível de significância agregado do teste T bicaudal
de estudantes, referente a uma série de valores. (weight, value)
```

ttest1w_sterr

TTest1w_sterr() retorna o erro padrão agregado da diferença média do teste T de estudantes, referente a uma série de valores.

```
TTest1w_sterr() retorna o erro padrão agregado da diferença média do teste T
de estudantes, referente a uma série de valores. (weight, value)
```

ttest1w_t

TTest1w_t() retorna o valor t agregado para um série de valores.

```
TTest1w_t() retorna o valor t agregado para um série de valores. ( weight,
value)
```

ttest1w_upper

TTest1w_upper() retorna o valor agregado da extremidade superior do intervalo de confiança, referente a uma série de valores.

```
TTest1w_upper() retorna o valor agregado da extremidade superior do intervalo
de confiança, referente a uma série de valores. (weight, value [, sig])
```

TTest_conf

TTest_conf retorna o valor agregado do intervalo de confiança do teste T para duas amostras independentes.

Essa função se aplica a testes T de Student com amostras independentes.

Se a função for usada no script de carga dos dados, os valores serão iterados com base em um número de registros que são definidos por uma cláusula group by.

Se a função for usada para expressões de gráficos, os valores serão iterados com base nas dimensões do gráfico.

Sintaxe:

```
TTest_conf ( grp, value [, sig [, eq_var]])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
value	Os valores da amostra a serem avaliados. Os valores da amostra devem ser logicamente agrupados, como especificado por exatamente dois valores em **group**. Se um nome de campo dos valores das amostras não for fornecido no script de carregamento, o campo será automaticamente nomeado como **Value**.
grp	O campo que contém os nomes de cada um dos dois grupos de amostras. Se um nome de campo para o grupo não for fornecido no script de carregamento, o campo será automaticamente nomeado como **Type**.
sig	É possível especificar o nível de significância bicaudal em **sig**. Se omitido, **sig** será definido como 0,025, gerando um intervalo de confiança de 95%.
eq_var	Se **eq_var** for especificado como False (0), variâncias separadas das duas amostras serão assumidas. Se **eq_var** for especificado como True (1), variâncias iguais entre as duas amostras serão assumidas.

Limitações:

Valores de texto, valores NULL e valores ausentes nas expressões de valores farão com que a função retorne um resultado NULL.

Exemplos:

```
TTest_conf( Group, Value )
TTest_conf( Group, Value, Sig, false )
```

Consulte também:

Criando um relatório típico de t-test (page 530)

TTest_df

TTest_df() retorna o valor (graus de liberdade) agregado do teste T de estudante referente a duas séries independentes de valores.

Essa função se aplica a testes T de Student com amostras independentes.

Se a função for usada no script de carga dos dados, os valores serão iterados com base em um número de registros que são definidos por uma cláusula group by.

Se a função for usada para expressões de gráficos, os valores serão iterados com base nas dimensões do gráfico.

Sintaxe:

```
TTest_df (grp, value [, eq_var])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
value	Os valores da amostra a serem avaliados. Os valores da amostra devem ser logicamente agrupados, como especificado por exatamente dois valores em **group**. Se um nome de campo dos valores das amostras não for fornecido no script de carregamento, o campo será automaticamente nomeado como **Value**.
grp	O campo que contém os nomes de cada um dos dois grupos de amostras. Se um nome de campo para o grupo não for fornecido no script de carregamento, o campo será automaticamente nomeado como **Type**.
eq_var	Se **eq_var** for especificado como False (0), variâncias separadas das duas amostras serão assumidas. Se **eq_var** for especificado como True (1), variâncias iguais entre as duas amostras serão assumidas.

Limitações:

Valores de texto, valores NULL e valores ausentes nas expressões de valores farão com que a função retorne um resultado NULL.

Exemplos:

```
TTest_df( Group, Value )
TTest_df( Group, Value, false )
```

Consulte também:

Criando um relatório típico de t-test (page 530)

TTest_dif

TTest_dif() é uma função numérica que retorna a diferença média agregada do teste T de estudantes, referente a duas séries independentes de valores.

Essa função se aplica a testes T de Student com amostras independentes.

Se a função for usada no script de carga dos dados, os valores serão iterados com base em um número de registros que são definidos por uma cláusula group by.

Se a função for usada para expressões de gráficos, os valores serão iterados com base nas dimensões do gráfico.

Sintaxe:

```
TTest_dif (grp, value [, eq_var] )
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
value	Os valores da amostra a serem avaliados. Os valores da amostra devem ser logicamente agrupados, como especificado por exatamente dois valores em **group**. Se um nome de campo dos valores das amostras não for fornecido no script de carregamento, o campo será automaticamente nomeado como **Value**.
grp	O campo que contém os nomes de cada um dos dois grupos de amostras. Se um nome de campo para o grupo não for fornecido no script de carregamento, o campo será automaticamente nomeado como **Type**.
eq_var	Se **eq_var** for especificado como False (0), variâncias separadas das duas amostras serão assumidas. Se **eq_var** for especificado como True (1), variâncias iguais entre as duas amostras serão assumidas.

Limitações:

Valores de texto, valores NULL e valores ausentes nas expressões de valores farão com que a função retorne um resultado NULL.

Exemplos:

```
TTest_dif( Group, Value )
TTest_dif( Group, Value, false )
```

Consulte também:

Criando um relatório típico de t-test (page 530)

TTest_lower

TTest_lower() retorna o valor agregado da extremidade inferior do intervalo de confiança, referente a duas séries independentes de valores.

Essa função se aplica a testes T de Student com amostras independentes.

Se a função for usada no script de carga dos dados, os valores serão iterados com base em um número de registros que são definidos por uma cláusula group by.

Se a função for usada para expressões de gráficos, os valores serão iterados com base nas dimensões do gráfico.

Sintaxe:

```
TTest_lower (grp, value [, sig [, eq_var]])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
value	Os valores da amostra a serem avaliados. Os valores da amostra devem ser logicamente agrupados, como especificado por exatamente dois valores em **group**. Se um nome de campo dos valores das amostras não for fornecido no script de carregamento, o campo será automaticamente nomeado como **Value**.
grp	O campo que contém os nomes de cada um dos dois grupos de amostras. Se um nome de campo para o grupo não for fornecido no script de carregamento, o campo será automaticamente nomeado como **Type**.
sig	É possível especificar o nível de significância bicaudal em **sig**. Se omitido, **sig** será definido como 0,025, gerando um intervalo de confiança de 95%.
eq_var	Se **eq_var** for especificado como False (0), variâncias separadas das duas amostras serão assumidas. Se **eq_var** for especificado como True (1), variâncias iguais entre as duas amostras serão assumidas.

Limitações:

Valores de texto, valores NULL e valores ausentes nas expressões de valores farão com que a função retorne um resultado NULL.

Exemplos:

```
TTest_lower( Group, Value )
TTest_lower( Group, Value, Sig, false )
```

Consulte também:

Criando um relatório típico de t-test (page 530)

TTest_sig

TTest_sig() retorna o nível de significância agregado do teste T bicaudal de estudantes, referente a duas séries independentes de valores.

Essa função se aplica a testes T de Student com amostras independentes.

Se a função for usada no script de carga dos dados, os valores serão iterados com base em um número de registros que são definidos por uma cláusula group by.

Se a função for usada para expressões de gráficos, os valores serão iterados com base nas dimensões do gráfico.

Sintaxe:

```
TTest_sig (grp, value [, eq_var])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
value	Os valores da amostra a serem avaliados. Os valores da amostra devem ser logicamente agrupados, como especificado por exatamente dois valores em **group**. Se um nome de campo dos valores das amostras não for fornecido no script de carregamento, o campo será automaticamente nomeado como **Value**.
grp	O campo que contém os nomes de cada um dos dois grupos de amostras. Se um nome de campo para o grupo não for fornecido no script de carregamento, o campo será automaticamente nomeado como **Type**.
eq_var	Se **eq_var** for especificado como False (0), variâncias separadas das duas amostras serão assumidas. Se **eq_var** for especificado como True (1), variâncias iguais entre as duas amostras serão assumidas.

Limitações:

Valores de texto, valores NULL e valores ausentes nas expressões de valores farão com que a função retorne um resultado NULL.

Exemplos:

```
TTest_sig( Group, Value )
TTest_sig( Group, Value, false )
```

Consulte também:

Criando um relatório típico de t-test (page 530)

TTest_sterr

TTest_sterr() retorna o erro padrão agregado da diferença média do teste T de estudantes, referente a duas séries independentes de valores.

Essa função se aplica a testes T de Student com amostras independentes.

Se a função for usada no script de carga dos dados, os valores serão iterados com base em um número de registros que são definidos por uma cláusula group by.

Se a função for usada para expressões de gráficos, os valores serão iterados com base nas dimensões do gráfico.

Sintaxe:

```
TTest_sterr (grp, value [, eq_var])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
value	Os valores da amostra a serem avaliados. Os valores da amostra devem ser logicamente agrupados, como especificado por exatamente dois valores em **group**. Se um nome de campo dos valores das amostras não for fornecido no script de carregamento, o campo será automaticamente nomeado como **Value**.
grp	O campo que contém os nomes de cada um dos dois grupos de amostras. Se um nome de campo para o grupo não for fornecido no script de carregamento, o campo será automaticamente nomeado como **Type**.
eq_var	Se **eq_var** for especificado como False (0), variâncias separadas das duas amostras serão assumidas. Se **eq_var** for especificado como True (1), variâncias iguais entre as duas amostras serão assumidas.

Limitações:

Valores de texto, valores NULL e valores ausentes nas expressões de valores farão com que a função retorne um resultado NULL.

Exemplos:

```
TTest_sterr( Group, Value )
TTest_sterr( Group, Value, false )
```

Consulte também:

Criando um relatório típico de t-test (page 530)

TTest_t

TTest_t() retorna o valor t agregado referente a duas séries independentes de valores.

Essa função se aplica a testes T de Student com amostras independentes.

Se a função for usada no script de carga dos dados, os valores serão iterados com base em um número de registros que são definidos por uma cláusula group by.

Se a função for usada para expressões de gráficos, os valores serão iterados com base nas dimensões do gráfico.

Sintaxe:

```
TTest_t(grp, value[, eq_var])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
value	Os valores da amostra a serem avaliados. Os valores da amostra devem ser logicamente agrupados, como especificado por exatamente dois valores em **group**. Se um nome de campo dos valores das amostras não for fornecido no script de carregamento, o campo será automaticamente nomeado como **Value**.
grp	O campo que contém os nomes de cada um dos dois grupos de amostras. Se um nome de campo para o grupo não for fornecido no script de carregamento, o campo será automaticamente nomeado como **Type**.
eq_var	Se **eq_var** for especificado como False (0), variâncias separadas das duas amostras serão assumidas. Se **eq_var** for especificado como True (1), variâncias iguais entre as duas amostras serão assumidas.

Limitações:

Valores de texto, valores NULL e valores ausentes nas expressões de valores farão com que a função retorne um resultado NULL.

Exemplo:

```
TTest_t( Group, Value, false )
```

Consulte também:

Criando um relatório típico de t-test (page 530)

TTest_upper

TTest_upper() retorna o valor agregado da extremidade superior do intervalo de confiança, referente a duas séries independentes de valores.

Essa função se aplica a testes T de Student com amostras independentes.

Se a função for usada no script de carga dos dados, os valores serão iterados com base em um número de registros que são definidos por uma cláusula group by.

Se a função for usada para expressões de gráficos, os valores serão iterados com base nas dimensões do gráfico.

Sintaxe:

```
TTest_upper (grp, value [, sig [, eq_var]])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
value	Os valores da amostra a serem avaliados. Os valores da amostra devem ser logicamente agrupados, como especificado por exatamente dois valores em **group**. Se um nome de campo dos valores das amostras não for fornecido no script de carregamento, o campo será automaticamente nomeado como **Value**.
grp	O campo que contém os nomes de cada um dos dois grupos de amostras. Se um nome de campo para o grupo não for fornecido no script de carregamento, o campo será automaticamente nomeado como **Type**.
sig	É possível especificar o nível de significância bicaudal em **sig**. Se omitido, **sig** será definido como 0,025, gerando um intervalo de confiança de 95%.
eq_var	Se **eq_var** for especificado como False (0), variâncias separadas das duas amostras serão assumidas. Se **eq_var** for especificado como True (1), variâncias iguais entre as duas amostras serão assumidas.

Limitações:

Valores de texto, valores NULL e valores ausentes nas expressões de valores farão com que a função retorne um resultado NULL.

Exemplos:

```
TTest_upper( Group, Value )
TTest_upper( Group, Value, sig, false )
```

Consulte também:

Criando um relatório típico de t-test (page 530)

TTestw_conf

TTestw_conf() retorna o valor t agregado referente a duas séries independentes de valores.

Essa função se aplica a testes T de Student com amostras independentes em que a série de dados de entrada é fornecida em um formato de duas colunas ponderadas.

Se a função for usada no script de carga dos dados, os valores serão iterados com base em um número de registros que são definidos por uma cláusula group by.

Se a função for usada para expressões de gráficos, os valores serão iterados com base nas dimensões do gráfico.

Sintaxe:

```
TTestw_conf (weight, grp, value [, sig [, eq_var]])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
value	Os valores da amostra a serem avaliados. Os valores da amostra devem ser logicamente agrupados, como especificado por exatamente dois valores em **group**. Se um nome de campo dos valores das amostras não for fornecido no script de carregamento, o campo será automaticamente nomeado como **Value**.
weight	Cada valor em **value** pode ser calculado uma ou mais vezes, de acordo com um valor de ponderação correspondente em **weight**.
grp	O campo que contém os nomes de cada um dos dois grupos de amostras. Se um nome de campo para o grupo não for fornecido no script de carregamento, o campo será automaticamente nomeado como **Type**.
sig	É possível especificar o nível de significância bicaudal em **sig**. Se omitido, **sig** será definido como 0,025, gerando um intervalo de confiança de 95%.
eq_var	Se **eq_var** for especificado como False (0), variâncias separadas das duas amostras serão assumidas. Se **eq_var** for especificado como True (1), variâncias iguais entre as duas amostras serão assumidas.

Limitações:

Valores de texto, valores NULL e valores ausentes nas expressões de valores farão com que a função retorne um resultado NULL.

Exemplos:

```
TTestw_conf( Weight, Group, Value )
TTestw_conf( Weight, Group, Value, sig, false )
```

Consulte também:

Criando um relatório típico de t-test (page 530)

TTestw_df

TTestw_df() retorna o valor df (graus de liberdade) agregado do teste T de estudante referente a duas séries independentes de valores.

Essa função se aplica a testes T de Student com amostras independentes em que a série de dados de entrada é fornecida em um formato de duas colunas ponderadas.

Se a função for usada no script de carga dos dados, os valores serão iterados com base em um número de registros que são definidos por uma cláusula group by.

Se a função for usada para expressões de gráficos, os valores serão iterados com base nas dimensões do gráfico.

Sintaxe:

```
TTestw_df (weight, grp, value [, eq_var])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
weight	Cada valor em **value** pode ser calculado uma ou mais vezes, de acordo com um valor de ponderação correspondente em **weight**.
grp	O campo que contém os nomes de cada um dos dois grupos de amostras. Se um nome de campo para o grupo não for fornecido no script de carregamento, o campo será automaticamente nomeado como **Type**.
value	Os valores da amostra a serem avaliados. Os valores da amostra devem ser logicamente agrupados, como especificado por exatamente dois valores em **group**. Se um nome de campo dos valores das amostras não for fornecido no script de carregamento, o campo será automaticamente nomeado como **Value**.
eq_var	Se **eq_var** for especificado como False (0), variâncias separadas das duas amostras serão assumidas. Se **eq_var** for especificado como True (1), variâncias iguais entre as duas amostras serão assumidas.

Limitações:

Valores de texto, valores NULL e valores ausentes nas expressões de valores farão com que a função retorne um resultado NULL.

Exemplos:

```
TTestw_df( Weight, Group, Value )
TTestw_df( Weight, Group, Value, false )
```

Consulte também:

Criando um relatório típico de t-test (page 530)

TTestw_dif

TTestw_dif() retorna a diferença média agregada do teste T de estudantes, referente a duas séries independentes de valores.

Essa função se aplica a testes T de Student com amostras independentes em que a série de dados de entrada é fornecida em um formato de duas colunas ponderadas.

Se a função for usada no script de carga dos dados, os valores serão iterados com base em um número de registros que são definidos por uma cláusula group by.

Se a função for usada para expressões de gráficos, os valores serão iterados com base nas dimensões do gráfico.

Sintaxe:

```
TTestw_dif (weight, grp, value)
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
weight	Cada valor em **value** pode ser calculado uma ou mais vezes, de acordo com um valor de ponderação correspondente em **weight**.
grp	O campo que contém os nomes de cada um dos dois grupos de amostras. Se um nome de campo para o grupo não for fornecido no script de carregamento, o campo será automaticamente nomeado como **Type**.
value	Os valores da amostra a serem avaliados. Os valores da amostra devem ser logicamente agrupados, como especificado por exatamente dois valores em **group**. Se um nome de campo dos valores das amostras não for fornecido no script de carregamento, o campo será automaticamente nomeado como **Value**.

Limitações:

Valores de texto, valores NULL e valores ausentes nas expressões de valores farão com que a função retorne um resultado NULL.

Exemplos:

```
TTestw_dif( Weight, Group, Value )
TTestw_dif( Weight, Group, Value, false )
```

Consulte também:

Criando um relatório típico de t-test (page 530)

TTestw_lower

TTestw_lower() retorna o valor agregado da extremidade inferior do intervalo de confiança, referente a duas séries independentes de valores.

Essa função se aplica a testes T de Student com amostras independentes em que a série de dados de entrada é fornecida em um formato de duas colunas ponderadas.

Se a função for usada no script de carga dos dados, os valores serão iterados com base em um número de registros que são definidos por uma cláusula group by.

Se a função for usada para expressões de gráficos, os valores serão iterados com base nas dimensões do gráfico.

Sintaxe:

```
TTestw_lower (weight, grp, value [, sig [, eq_var]])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
weight	Cada valor em **value** pode ser calculado uma ou mais vezes, de acordo com um valor de ponderação correspondente em **weight**.
grp	O campo que contém os nomes de cada um dos dois grupos de amostras. Se um nome de campo para o grupo não for fornecido no script de carregamento, o campo será automaticamente nomeado como **Type**.
value	Os valores da amostra a serem avaliados. Os valores da amostra devem ser logicamente agrupados, como especificado por exatamente dois valores em **group**. Se um nome de campo dos valores das amostras não for fornecido no script de carregamento, o campo será automaticamente nomeado como **Value**.
sig	É possível especificar o nível de significância bicaudal em **sig**. Se omitido, **sig** será definido como 0,025, gerando um intervalo de confiança de 95%.
eq_var	Se **eq_var** for especificado como False (0), variâncias separadas das duas amostras serão assumidas. Se **eq_var** for especificado como True (1), variâncias iguais entre as duas amostras serão assumidas.

Limitações:

Valores de texto, valores NULL e valores ausentes nas expressões de valores farão com que a função retorne um resultado NULL.

Exemplos:

```
TTestw_lower( Weight, Group, Value )
TTestw_lower( Weight, Group, Value, sig, false )
```

Consulte também:

Criando um relatório típico de t-test (page 530)

TTestw_sig

TTestw_sig() retorna o nível de significância agregado do teste T bicaudal de estudantes, referente a duas séries independentes de valores.

Essa função se aplica a testes T de Student com amostras independentes em que a série de dados de entrada é fornecida em um formato de duas colunas ponderadas.

Se a função for usada no script de carga dos dados, os valores serão iterados com base em um número de registros que são definidos por uma cláusula group by.

Se a função for usada para expressões de gráficos, os valores serão iterados com base nas dimensões do gráfico.

Sintaxe:

```
TTestw_sig ( weight, grp, value [, eq_var])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
weight	Cada valor em **value** pode ser calculado uma ou mais vezes, de acordo com um valor de ponderação correspondente em **weight**.
grp	O campo que contém os nomes de cada um dos dois grupos de amostras. Se um nome de campo para o grupo não for fornecido no script de carregamento, o campo será automaticamente nomeado como **Type**.
value	Os valores da amostra a serem avaliados. Os valores da amostra devem ser logicamente agrupados, como especificado por exatamente dois valores em **group**. Se um nome de campo dos valores das amostras não for fornecido no script de carregamento, o campo será automaticamente nomeado como **Value**.
eq_var	Se **eq_var** for especificado como False (0), variâncias separadas das duas amostras serão assumidas. Se **eq_var** for especificado como True (1), variâncias iguais entre as duas amostras serão assumidas.

Limitações:

Valores de texto, valores NULL e valores ausentes nas expressões de valores farão com que a função retorne um resultado NULL.

Exemplos:

```
TTestw_sig( Weight, Group, Value )
TTestw_sig( Weight, Group, Value, false )
```

Consulte também:

Criando um relatório típico de t-test (page 530)

TTestw_sterr

TTestw_sterr() retorna o erro padrão agregado da diferença média do teste T de estudantes, referente a duas séries independentes de valores.

Essa função se aplica a testes T de Student com amostras independentes em que a série de dados de entrada é fornecida em um formato de duas colunas ponderadas.

Se a função for usada no script de carga dos dados, os valores serão iterados com base em um número de registros que são definidos por uma cláusula group by.

Se a função for usada para expressões de gráficos, os valores serão iterados com base nas dimensões do gráfico.

Sintaxe:

```
TTestw_sterr (weight, grp, value [, eq_var])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
weight	Cada valor em **value** pode ser calculado uma ou mais vezes, de acordo com um valor de ponderação correspondente em **weight**.
grp	O campo que contém os nomes de cada um dos dois grupos de amostras. Se um nome de campo para o grupo não for fornecido no script de carregamento, o campo será automaticamente nomeado como **Type**.
value	Os valores da amostra a serem avaliados. Os valores da amostra devem ser logicamente agrupados, como especificado por exatamente dois valores em **group**. Se um nome de campo dos valores das amostras não for fornecido no script de carregamento, o campo será automaticamente nomeado como **Value**.
eq_var	Se **eq_var** for especificado como False (0), variâncias separadas das duas amostras serão assumidas. Se **eq_var** for especificado como True (1), variâncias iguais entre as duas amostras serão assumidas.

Limitações:

Valores de texto, valores NULL e valores ausentes nas expressões de valores farão com que a função retorne um resultado NULL.

Exemplos:

```
TTestw_sterr( Weight, Group, Value )
TTestw_sterr( Weight, Group, Value, false )
```

Consulte também:

Criando um relatório típico de t-test (page 530)

TTestw_t

TTestw_t() retorna o valor t agregado referente a duas séries independentes de valores.

Essa função se aplica a testes T de Student com amostras independentes em que a série de dados de entrada é fornecida em um formato de duas colunas ponderadas.

Se a função for usada no script de carga dos dados, os valores serão iterados com base em um número de registros que são definidos por uma cláusula group by.

Se a função for usada para expressões de gráficos, os valores serão iterados com base nas dimensões do gráfico.

Sintaxe:

```
ttestw_t (weight, grp, value [, eq_var])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
value	Os valores da amostra a serem avaliados. Os valores da amostra devem ser logicamente agrupados, como especificado por exatamente dois valores em **group**. Se um nome de campo dos valores das amostras não for fornecido no script de carregamento, o campo será automaticamente nomeado como **Value**.
weight	Cada valor em **value** pode ser calculado uma ou mais vezes, de acordo com um valor de ponderação correspondente em **weight**.
grp	O campo que contém os nomes de cada um dos dois grupos de amostras. Se um nome de campo para o grupo não for fornecido no script de carregamento, o campo será automaticamente nomeado como **Type**.

Argumento	Descrição
eq_var	Se **eq_var** for especificado como False (0), variâncias separadas das duas amostras serão assumidas. Se **eq_var** for especificado como True (1), variâncias iguais entre as duas amostras serão assumidas.

Limitações:

Valores de texto, valores NULL e valores ausentes nas expressões de valores farão com que a função retorne um resultado NULL.

Exemplos:

```
TTestw_t( Weight, Group, Value )
TTestw_t( Weight, Group, Value, false )
```

Consulte também:

Criando um relatório típico de t-test (page 530)

TTestw_upper

TTestw_upper() retorna o valor agregado da extremidade superior do intervalo de confiança, referente a duas séries independentes de valores.

Essa função se aplica a testes T de Student com amostras independentes em que a série de dados de entrada é fornecida em um formato de duas colunas ponderadas.

Se a função for usada no script de carga dos dados, os valores serão iterados com base em um número de registros que são definidos por uma cláusula group by.

Se a função for usada para expressões de gráficos, os valores serão iterados com base nas dimensões do gráfico.

Sintaxe:

```
TTestw_upper (weight, grp, value [, sig [, eq_var]])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
weight	Cada valor em **value** pode ser calculado uma ou mais vezes, de acordo com um valor de ponderação correspondente em **weight**.

Argumento	Descrição
grp	O campo que contém os nomes de cada um dos dois grupos de amostras. Se um nome de campo para o grupo não for fornecido no script de carregamento, o campo será automaticamente nomeado como **Type**.
value	Os valores da amostra a serem avaliados. Os valores da amostra devem ser logicamente agrupados, como especificado por exatamente dois valores em **group**. Se um nome de campo dos valores das amostras não for fornecido no script de carregamento, o campo será automaticamente nomeado como **Value**.
sig	É possível especificar o nível de significância bicaudal em **sig**. Se omitido, **sig** será definido como 0,025, gerando um intervalo de confiança de 95%.
eq_var	Se **eq_var** for especificado como False (0), variâncias separadas das duas amostras serão assumidas. Se **eq_var** for especificado como True (1), variâncias iguais entre as duas amostras serão assumidas.

Limitações:

Valores de texto, valores NULL e valores ausentes nas expressões de valores farão com que a função retorne um resultado NULL.

Exemplos:

```
TTestw_upper( Weight, Group, Value )
TTestw_upper( Weight, Group, Value, sig, false )
```

Consulte também:

Criando um relatório típico de t-test (page 530)

TTest1_conf

TTest1_conf() retorna o valor de intervalo de confiança agregado referente a uma série de valores.

Essa função se aplica a testes T de Student com uma amostra.

Se a função for usada no script de carga dos dados, os valores serão iterados com base em um número de registros que são definidos por uma cláusula group by.

Se a função for usada para expressões de gráficos, os valores serão iterados com base nas dimensões do gráfico.

Sintaxe:

```
TTest1_conf (value [, sig ])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
value	As amostras a serem avaliadas. Se um nome de campo dos valores das amostras não for fornecido no script de carregamento, o campo será automaticamente nomeado como **Value**.
sig	É possível especificar o nível de significância bicaudal em **sig**. Se omitido, **sig** será definido como 0,025, gerando um intervalo de confiança de 95%.

Limitações:

Valores de texto, valores NULL e valores ausentes nas expressões de valores farão com que a função retorne um resultado NULL.

Exemplos:

```
TTest1_conf( Value )
TTest1_conf( Value, 0.005 )
```

Consulte também:

Criando um relatório típico de t-test (page 530)

TTest1_df

TTest1_df() retorna o valor df (graus de liberdade) agregado do teste T de estudante referente a uma série de valores.

Essa função se aplica a testes T de Student com uma amostra.

Se a função for usada no script de carga dos dados, os valores serão iterados com base em um número de registros que são definidos por uma cláusula group by.

Se a função for usada para expressões de gráficos, os valores serão iterados com base nas dimensões do gráfico.

Sintaxe:

```
TTest1_df (value)
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
value	As amostras a serem avaliadas. Se um nome de campo dos valores das amostras não for fornecido no script de carregamento, o campo será automaticamente nomeado como **Value**.

Limitações:

Valores de texto, valores NULL e valores ausentes nas expressões de valores farão com que a função retorne um resultado NULL.

Exemplo:

```
TTest1_df( Value )
```

Consulte também:

Criando um relatório típico de t-test (page 530)

TTest1_dif

TTest1_dif() retorna a diferença média agregada do teste T de estudantes, referente a uma série de valores.

Essa função se aplica a testes T de Student com uma amostra.

Se a função for usada no script de carga dos dados, os valores serão iterados com base em um número de registros que são definidos por uma cláusula group by.

Se a função for usada para expressões de gráficos, os valores serão iterados com base nas dimensões do gráfico.

Sintaxe:

```
TTest1_dif (value)
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
value	As amostras a serem avaliadas. Se um nome de campo dos valores das amostras não for fornecido no script de carregamento, o campo será automaticamente nomeado como **Value**.

Limitações:

Valores de texto, valores NULL e valores ausentes nas expressões de valores farão com que a função retorne um resultado NULL.

Exemplo:

```
TTest1_dif( Value )
```

Consulte também:

Criando um relatório típico de t-test (page 530)

TTest1_lower

TTest1_lower() retorna o valor agregado da extremidade inferior do intervalo de confiança, referente a uma série de valores.

Essa função se aplica a testes T de Student com uma amostra.

Se a função for usada no script de carga dos dados, os valores serão iterados com base em um número de registros que são definidos por uma cláusula group by.

Se a função for usada para expressões de gráficos, os valores serão iterados com base nas dimensões do gráfico.

Sintaxe:

```
TTest1_lower (value [, sig])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
value	As amostras a serem avaliadas. Se um nome de campo dos valores das amostras não for fornecido no script de carregamento, o campo será automaticamente nomeado como **Value**.
sig	É possível especificar o nível de significância bicaudal em **sig**. Se omitido, **sig** será definido como 0,025, gerando um intervalo de confiança de 95%.

Limitações:

Valores de texto, valores NULL e valores ausentes nas expressões de valores farão com que a função retorne um resultado NULL.

Exemplos:

```
TTest1_lower( value )
TTest1_lower( value, 0.005 )
```

Consulte também:

Criando um relatório típico de t-test (page 530)

TTest1_sig

TTest1_sig() retorna o nível de significância agregado do teste T bicaudal de estudantes, referente a uma série de valores.

Essa função se aplica a testes T de Student com uma amostra.

Se a função for usada no script de carga dos dados, os valores serão iterados com base em um número de registros que são definidos por uma cláusula group by.

Se a função for usada para expressões de gráficos, os valores serão iterados com base nas dimensões do gráfico.

Sintaxe:

```
TTest1_sig (value)
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
value	As amostras a serem avaliadas. Se um nome de campo dos valores das amostras não for fornecido no script de carregamento, o campo será automaticamente nomeado como **Value**.

Limitações:

Valores de texto, valores NULL e valores ausentes nas expressões de valores farão com que a função retorne um resultado NULL.

Exemplo:

```
TTest1_sig( Value )
```

Consulte também:

Criando um relatório típico de t-test (page 530)

TTest1_sterr

TTest1_sterr() retorna o erro padrão agregado da diferença média do teste T de estudantes, referente a uma série de valores.

Essa função se aplica a testes T de Student com uma amostra.

Se a função for usada no script de carga dos dados, os valores serão iterados com base em um número de registros que são definidos por uma cláusula group by.

Se a função for usada para expressões de gráficos, os valores serão iterados com base nas dimensões do gráfico.

Sintaxe:

```
TTest1_sterr (value)
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
value	As amostras a serem avaliadas. Se um nome de campo dos valores das amostras não for fornecido no script de carregamento, o campo será automaticamente nomeado como **Value**.

Limitações:

Valores de texto, valores NULL e valores ausentes nas expressões de valores farão com que a função retorne um resultado NULL.

Exemplo:

```
TTest1_sterr( Value )
```

Consulte também:

Criando um relatório típico de t-test (page 530)

TTest1_t

TTest1_t() retorna o valor t agregado para um série de valores.

Essa função se aplica a testes T de Student com uma amostra.

Se a função for usada no script de carga dos dados, os valores serão iterados com base em um número de registros que são definidos por uma cláusula group by.

Se a função for usada para expressões de gráficos, os valores serão iterados com base nas dimensões do gráfico.

Sintaxe:

```
TTest1_t (value)
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
value	As amostras a serem avaliadas. Se um nome de campo dos valores das amostras não for fornecido no script de carregamento, o campo será automaticamente nomeado como **Value**.

Limitações:

Valores de texto, valores NULL e valores ausentes nas expressões de valores farão com que a função retorne um resultado NULL.

Exemplo:

```
TTest1_t( Value )
```

Consulte também:

Criando um relatório típico de t-test (page 530)

TTest1_upper

TTest1_upper() retorna o valor agregado da extremidade superior do intervalo de confiança, referente a uma série de valores.

Essa função se aplica a testes T de Student com uma amostra.

Se a função for usada no script de carga dos dados, os valores serão iterados com base em um número de registros que são definidos por uma cláusula group by.

Se a função for usada para expressões de gráficos, os valores serão iterados com base nas dimensões do gráfico.

Sintaxe:

```
TTest1_upper (value [, sig])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
value	As amostras a serem avaliadas. Se um nome de campo dos valores das amostras não for fornecido no script de carregamento, o campo será automaticamente nomeado como **Value**.
sig	É possível especificar o nível de significância bicaudal em **sig**. Se omitido, **sig** será definido como 0,025, gerando um intervalo de confiança de 95%.

Limitações:

Valores de texto, valores NULL e valores ausentes nas expressões de valores farão com que a função retorne um resultado NULL.

Exemplos:

```
TTest1_upper( Value )
TTest1_upper( Value, 0.005 )
```

Consulte também:

Criando um relatório típico de t-test (page 530)

TTest1w_conf

TTest1w_conf() é uma função **numérica** que retorna o valor de intervalo de confiança agregado referente a uma série de valores.

Essa função se aplica a testes T de Student com uma amostra em que a série de dados de entrada é fornecida em um formato de duas colunas ponderadas.

Se a função for usada no script de carga dos dados, os valores serão iterados com base em um número de registros que são definidos por uma cláusula group by.

Se a função for usada para expressões de gráficos, os valores serão iterados com base nas dimensões do gráfico.

Sintaxe:

```
TTest1w_conf (weight, value [, sig ])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
value	As amostras a serem avaliadas. Se um nome de campo dos valores das amostras não for fornecido no script de carregamento, o campo será automaticamente nomeado como **Value**.
weight	Cada valor em **value** pode ser calculado uma ou mais vezes, de acordo com um valor de ponderação correspondente em **weight**.
sig	É possível especificar o nível de significância bicaudal em **sig**. Se omitido, **sig** será definido como 0,025, gerando um intervalo de confiança de 95%.

Limitações:

Valores de texto, valores NULL e valores ausentes nas expressões de valores farão com que a função retorne um resultado NULL.

Exemplos:

```
TTest1w_conf( Weight, Value )
TTest1w_conf( Weight, Value, 0.005 )
```

Consulte também:

Criando um relatório típico de t-test (page 530)

TTest1w_df

TTest1w_df() retorna o valor df (graus de liberdade) agregado do teste T de estudante referente a uma série de valores.

Essa função se aplica a testes T de Student com uma amostra em que a série de dados de entrada é fornecida em um formato de duas colunas ponderadas.

Se a função for usada no script de carga dos dados, os valores serão iterados com base em um número de registros que são definidos por uma cláusula group by.

Se a função for usada para expressões de gráficos, os valores serão iterados com base nas dimensões do gráfico.

Sintaxe:

```
TTest1w_df (weight, value)
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
value	As amostras a serem avaliadas. Se um nome de campo dos valores das amostras não for fornecido no script de carregamento, o campo será automaticamente nomeado como **Value**.
weight	Cada valor em **value** pode ser calculado uma ou mais vezes, de acordo com um valor de ponderação correspondente em **weight**.

Limitações:

Valores de texto, valores NULL e valores ausentes nas expressões de valores farão com que a função retorne um resultado NULL.

Exemplo:

```
TTest1w_df( Weight, Value )
```

Consulte também:

Criando um relatório típico de t-test (page 530)

TTest1w_dif

TTest1w_dif() retorna a diferença média agregada do teste T de estudantes, referente a uma série de valores.

Essa função se aplica a testes T de Student com uma amostra em que a série de dados de entrada é fornecida em um formato de duas colunas ponderadas.

Se a função for usada no script de carga dos dados, os valores serão iterados com base em um número de registros que são definidos por uma cláusula group by.

Se a função for usada para expressões de gráficos, os valores serão iterados com base nas dimensões do gráfico.

Sintaxe:

```
TTest1w_dif (weight, value)
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
value	As amostras a serem avaliadas. Se um nome de campo dos valores das amostras não for fornecido no script de carregamento, o campo será automaticamente nomeado como **Value**.
weight	Cada valor em **value** pode ser calculado uma ou mais vezes, de acordo com um valor de ponderação correspondente em **weight**.

Limitações:

Valores de texto, valores NULL e valores ausentes nas expressões de valores farão com que a função retorne um resultado NULL.

Exemplo:

```
TTest1w_dif( Weight, Value )
```

Consulte também:

Criando um relatório típico de t-test (page 530)

TTest1w_lower

TTest1w_lower() retorna o valor agregado da extremidade inferior do intervalo de confiança, referente a uma série de valores.

Essa função se aplica a testes T de Student com uma amostra em que a série de dados de entrada é fornecida em um formato de duas colunas ponderadas.

Se a função for usada no script de carga dos dados, os valores serão iterados com base em um número de registros que são definidos por uma cláusula group by.

Se a função for usada para expressões de gráficos, os valores serão iterados com base nas dimensões do gráfico.

Sintaxe:

```
TTest1w_lower (weight, value [, sig ])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
value	As amostras a serem avaliadas. Se um nome de campo dos valores das amostras não for fornecido no script de carregamento, o campo será automaticamente nomeado como **Value**.
weight	Cada valor em **value** pode ser calculado uma ou mais vezes, de acordo com um valor de ponderação correspondente em **weight**.
sig	É possível especificar o nível de significância bicaudal em **sig**. Se omitido, **sig** será definido como 0,025, gerando um intervalo de confiança de 95%.

Limitações:

Valores de texto, valores NULL e valores ausentes nas expressões de valores farão com que a função retorne um resultado NULL.

Exemplos:

```
TTest1w_lower( Weight, Value )
TTest1w_lower( Weight, Value, 0.005 )
```

Consulte também:

Criando um relatório típico de t-test (page 530)

TTest1w_sig

TTest1w_sig() retorna o nível de significância agregado do teste T bicaudal de estudantes, referente a uma série de valores.

Essa função se aplica a testes T de Student com uma amostra em que a série de dados de entrada é fornecida em um formato de duas colunas ponderadas.

Se a função for usada no script de carga dos dados, os valores serão iterados com base em um número de registros que são definidos por uma cláusula group by.

Se a função for usada para expressões de gráficos, os valores serão iterados com base nas dimensões do gráfico.

Sintaxe:

```
TTest1w_sig (weight, value)
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
value	As amostras a serem avaliadas. Se um nome de campo dos valores das amostras não for fornecido no script de carregamento, o campo será automaticamente nomeado como **Value**.
weight	Cada valor em **value** pode ser calculado uma ou mais vezes, de acordo com um valor de ponderação correspondente em **weight**.

Limitações:

Valores de texto, valores NULL e valores ausentes nas expressões de valores farão com que a função retorne um resultado NULL.

Exemplo:

```
TTest1w_sig( Weight, Value )
```

Consulte também:

Criando um relatório típico de t-test (page 530)

TTest1w_sterr

TTest1w_sterr() retorna o erro padrão agregado da diferença média do teste T de estudantes, referente a uma série de valores.

Essa função se aplica a testes T de Student com uma amostra em que a série de dados de entrada é fornecida em um formato de duas colunas ponderadas.

Se a função for usada no script de carga dos dados, os valores serão iterados com base em um número de registros que são definidos por uma cláusula group by.

Se a função for usada para expressões de gráficos, os valores serão iterados com base nas dimensões do gráfico.

Sintaxe:

```
TTest1w_sterr (weight, value)
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
value	As amostras a serem avaliadas. Se um nome de campo dos valores das amostras não for fornecido no script de carregamento, o campo será automaticamente nomeado como **Value**.
weight	Cada valor em **value** pode ser calculado uma ou mais vezes, de acordo com um valor de ponderação correspondente em **weight**.

Limitações:

Valores de texto, valores NULL e valores ausentes nas expressões de valores farão com que a função retorne um resultado NULL.

Exemplo:

```
TTest1w_sterr( Weight, Value )
```

Consulte também:

Criando um relatório típico de t-test (page 530)

TTest1w_t

TTest1w_t() retorna o valor t agregado para um série de valores.

Essa função se aplica a testes T de Student com uma amostra em que a série de dados de entrada é fornecida em um formato de duas colunas ponderadas.

Se a função for usada no script de carga dos dados, os valores serão iterados com base em um número de registros que são definidos por uma cláusula group by.

Se a função for usada para expressões de gráficos, os valores serão iterados com base nas dimensões do gráfico.

Sintaxe:

```
TTest1w_t ( weight, value)
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
value	As amostras a serem avaliadas. Se um nome de campo dos valores das amostras não for fornecido no script de carregamento, o campo será automaticamente nomeado como **Value**.
weight	Cada valor em **value** pode ser calculado uma ou mais vezes, de acordo com um valor de ponderação correspondente em **weight**.

Limitações:

Valores de texto, valores NULL e valores ausentes nas expressões de valores farão com que a função retorne um resultado NULL.

Exemplo:

```
TTest1w_t( Weight, Value )
```

Consulte também:

Criando um relatório típico de t-test (page 530)

TTest1w_upper

TTest1w_upper() retorna o valor agregado da extremidade superior do intervalo de confiança, referente a uma série de valores.

Essa função se aplica a testes T de Student com uma amostra em que a série de dados de entrada é fornecida em um formato de duas colunas ponderadas.

Se a função for usada no script de carga dos dados, os valores serão iterados com base em um número de registros que são definidos por uma cláusula group by.

Se a função for usada para expressões de gráficos, os valores serão iterados com base nas dimensões do gráfico.

Sintaxe:

```
TTest1w_upper (weight, value [, sig])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
value	As amostras a serem avaliadas. Se um nome de campo dos valores das amostras não for fornecido no script de carregamento, o campo será automaticamente nomeado como **Value**.
weight	Cada valor em **value** pode ser calculado uma ou mais vezes, de acordo com um valor de ponderação correspondente em **weight**.
sig	É possível especificar o nível de significância bicaudal em **sig**. Se omitido, **sig** será definido como 0,025, gerando um intervalo de confiança de 95%.

Limitações:

Valores de texto, valores NULL e valores ausentes nas expressões de valores farão com que a função retorne um resultado NULL.

Exemplos:

```
TTest1w_upper( Weight, Value )
TTest1w_upper( Weight, Value, 0.005 )
```

Consulte também:

Criando um relatório típico de t-test (page 530)

Funções de teste Z

Um exame estatístico de duas médias populacionais. Um teste z de duas amostras examina se duas amostras são diferentes. Normalmente, é usado quando duas distribuições normais têm variâncias conhecidas e quando um experimento usa um tamanho grande de amostra.

No teste Z, as funções estatísticas de teste são agrupadas de acordo com o tipo de série de dados de entrada que se aplica à função.

Se a função for usada no script de carga dos dados, os valores serão iterados com base em um número de registros que são definidos por uma cláusula group by.

Se a função for usada para expressões de gráficos, os valores serão iterados com base nas dimensões do gráfico.

Exemplos de como usar funções z-test (page 534)

Funções de formato de uma coluna

As funções a seguir aplicam-se a testes z com séries de dados de entrada simples.

ztest_conf

ZTest_conf() retorna o valor z agregado de duas séries de valores.

```
ZTest_conf() retorna o valor z agregado de duas séries de valores. (value [, sigma [, sig ])
```

ztest_dif

ZTest_dif() retorna a diferença média agregada do teste z referente a uma série de valores.

```
ZTest_dif() retorna a diferença média agregada do teste z referente a uma série de valores. (value [, sigma])
```

ztest_sig

ZTest_sig() retorna o nível de significância agregado do teste z bicaudal, referente a uma série de valores.

```
ZTest_sig() retorna o nível de significância agregado do teste z bicaudal, referente a uma série de valores. (value [, sigma])
```

ztest_sterr

ZTest_sterr() retorna o erro padrão agregado da diferença média do teste z, referente a uma série de valores.

```
ZTest_sterr() retorna o erro padrão agregado da diferença média do teste z, referente a uma série de valores. (value [, sigma])
```

ztest_z

ZTest_z() retorna o valor z agregado de duas séries de valores.

```
ZTest_z() retorna o valor z agregado de duas séries de valores. (value [, sigma])
```

ztest_lower

ZTest_lower() retorna o valor agregado da extremidade inferior do intervalo de confiança, referente a duas séries independentes de valores.

```
ZTest_lower() retorna o valor agregado da extremidade inferior do intervalo de confiança, referente a duas séries independentes de valores. (grp, value [, sig [, eq_var]])
```

ztest_upper

ZTest_upper() retorna o valor agregado da extremidade superior do intervalo de confiança, referente a duas séries independentes de valores.

```
ZTest_upper() retorna o valor agregado da extremidade superior do intervalo de confiança, referente a duas séries independentes de valores. (grp, value [, sig [, eq_var]])
```

Funções de formato de duas colunas ponderadas

As funções a seguir aplicam-se aos testes z, nos quais a série de dados de entrada é fornecida em um formato de duas colunas ponderadas.

ztestw_conf

ZTestw_conf() retorna o valor de intervalo de confiança z agregado referente a uma série de valores.

```
ZTestw_conf() retorna o valor de intervalo de confiança z agregado referente a uma série de valores. (weight, value [, sigma [, sig]])
```

ztestw_dif

ZTestw_dif() retorna a diferença média agregada do teste z referente a uma série de valores.

```
ZTestw_dif() retorna a diferença média agregada do teste z referente a uma série de valores.  (weight, value [, sigma])
```

ztestw_lower

ZTestw_lower() retorna o valor agregado da extremidade inferior do intervalo de confiança, referente a duas séries independentes de valores.

```
ZTestw_lower() retorna o valor agregado da extremidade inferior do intervalo de confiança, referente a duas séries independentes de valores. (weight, value [, sigma])
```

ztestw_sig

ZTestw_sig() retorna o nível de significância agregado do teste z bicaudal, referente a uma série de valores.

```
ZTestw_sig() retorna o nível de significância agregado do teste z bicaudal, referente a uma série de valores. (weight, value [, sigma])
```

ztestw_sterr

ZTestw_sterr() retorna o erro padrão agregado da diferença média do teste z, referente a uma série de valores.

```
ZTestw_sterr() retorna o erro padrão agregado da diferença média do teste z, referente a uma série de valores. (weight, value [, sigma])
```

ztestw_upper

ZTestw_upper() retorna o valor agregado da extremidade superior do intervalo de confiança, referente a duas séries independentes de valores.

```
ZTestw_upper() retorna o valor agregado da extremidade superior do intervalo de confiança, referente a duas séries independentes de valores. (weight, value [, sigma])
```

ztestw_z

ZTestw_z() retorna o valor z agregado de duas séries de valores.

```
ZTestw_z() retorna o valor z agregado de duas séries de valores. (weight, value [, sigma])
```

ZTest_z

ZTest_z() retorna o valor z agregado de duas séries de valores.

Se a função for usada no script de carga dos dados, os valores serão iterados com base em um número de registros que são definidos por uma cláusula group by.

Se a função for usada para expressões de gráficos, os valores serão iterados com base nas dimensões do gráfico.

Sintaxe:

```
ZTest_z(value[, sigma])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
value	Os valores da amostra a serem avaliados. Pressupõe-se uma média amostral de 0. Para que o teste seja realizado com base em uma média diferente, subtraia essa média valor dos valores de amostra.
sigma	Se for conhecido, o desvio padrão pode ser indicado em **sigma**. Se **sigma** for omitido, o desvio padrão de amostra real será utilizado.

Limitações:

Valores de texto, valores NULL e valores ausentes nas expressões de valores farão com que a função retorne um resultado NULL.

Exemplo:

```
ZTest_z( Value-TestValue )
```

Consulte também:

Exemplos de como usar funções z-test (page 534)

ZTest_sig

ZTest_sig() retorna o nível de significância agregado do teste z bicaudal, referente a uma série de valores.

Se a função for usada no script de carga dos dados, os valores serão iterados com base em um número de registros que são definidos por uma cláusula group by.

Se a função for usada para expressões de gráficos, os valores serão iterados com base nas dimensões do gráfico.

Sintaxe:

```
ZTest_sig(value[, sigma])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
value	Os valores da amostra a serem avaliados. Pressupõe-se uma média amostral de 0. Para que o teste seja realizado com base em uma média diferente, subtraia essa média valor dos valores de amostra.
sigma	Se for conhecido, o desvio padrão pode ser indicado em **sigma**. Se **sigma** for omitido, o desvio padrão de amostra real será utilizado.

Limitações:

Valores de texto, valores NULL e valores ausentes nas expressões de valores farão com que a função retorne um resultado NULL.

Exemplo:

```
ZTest_sig(Value-TestValue)
```

Consulte também:

Exemplos de como usar funções z-test (page 534)

ZTest_dif

ZTest_dif() retorna a diferença média agregada do teste z referente a uma série de valores.

Se a função for usada no script de carga dos dados, os valores serão iterados com base em um número de registros que são definidos por uma cláusula group by.

Se a função for usada para expressões de gráficos, os valores serão iterados com base nas dimensões do gráfico.

Sintaxe:

```
ZTest_dif(value[, sigma])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
value	Os valores da amostra a serem avaliados. Pressupõe-se uma média amostral de 0. Para que o teste seja realizado com base em uma média diferente, subtraia essa média valor dos valores de amostra.
sigma	Se for conhecido, o desvio padrão pode ser indicado em **sigma**. Se **sigma** for omitido, o desvio padrão de amostra real será utilizado.

Limitações:

Valores de texto, valores NULL e valores ausentes nas expressões de valores farão com que a função retorne um resultado NULL.

Exemplo:

```
ZTest_dif(Value-TestValue)
```

Consulte também:

Exemplos de como usar funções z-test (page 534)

ZTest_sterr

ZTest_sterr() retorna o erro padrão agregado da diferença média do teste z, referente a uma série de valores.

Se a função for usada no script de carga dos dados, os valores serão iterados com base em um número de registros que são definidos por uma cláusula group by.

Se a função for usada para expressões de gráficos, os valores serão iterados com base nas dimensões do gráfico.

Sintaxe:

```
ZTest_sterr(value[, sigma])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
value	Os valores da amostra a serem avaliados. Pressupõe-se uma média amostral de 0. Para que o teste seja realizado com base em uma média diferente, subtraia essa média valor dos valores de amostra.
sigma	Se for conhecido, o desvio padrão pode ser indicado em **sigma**. Se **sigma** for omitido, o desvio padrão de amostra real será utilizado.

Limitações:

Valores de texto, valores NULL e valores ausentes nas expressões de valores farão com que a função retorne um resultado NULL.

Exemplo:

```
ZTest_sterr(Value-TestValue)
```

Consulte também:

Exemplos de como usar funções z-test (page 534)

ZTest_conf

ZTest_conf() retorna o valor z agregado de duas séries de valores.

Se a função for usada no script de carga dos dados, os valores serão iterados com base em um número de registros que são definidos por uma cláusula group by.

Se a função for usada para expressões de gráficos, os valores serão iterados com base nas dimensões do gráfico.

Sintaxe:

```
ZTest_conf(value[, sigma[, sig]])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
value	Os valores da amostra a serem avaliados. Pressupõe-se uma média amostral de 0. Para que o teste seja realizado com base em uma média diferente, subtraia essa média valor dos valores de amostra.
sigma	Se for conhecido, o desvio padrão pode ser indicado em **sigma**. Se **sigma** for omitido, o desvio padrão de amostra real será utilizado.
sig	É possível especificar o nível de significância bicaudal em **sig**. Se omitido, **sig** será definido como 0,025, gerando um intervalo de confiança de 95%.

Limitações:

Valores de texto, valores NULL e valores ausentes nas expressões de valores farão com que a função retorne um resultado NULL.

Exemplo:

```
ZTest_conf(Value-TestValue)
```

Consulte também:

Exemplos de como usar funções z-test (page 534)

ZTest_lower

ZTest_lower() retorna o valor agregado da extremidade inferior do intervalo de confiança, referente a duas séries independentes de valores.

Se a função for usada no script de carga dos dados, os valores serão iterados com base em um número de registros que são definidos por uma cláusula group by.

Se a função for usada para expressões de gráficos, os valores serão iterados com base nas dimensões do gráfico.

Sintaxe:

```
ZTest_lower (grp, value [, sig [, eq_var]])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
value	Os valores da amostra a serem avaliados. Os valores da amostra devem ser logicamente agrupados, como especificado por exatamente dois valores em **group**. Se um nome de campo dos valores das amostras não for fornecido no script de carregamento, o campo será automaticamente nomeado como **Value**.
grp	O campo que contém os nomes de cada um dos dois grupos de amostras. Se um nome de campo para o grupo não for fornecido no script de carregamento, o campo será automaticamente nomeado como **Type**.
sig	É possível especificar o nível de significância bicaudal em **sig**. Se omitido, **sig** será definido como 0,025, gerando um intervalo de confiança de 95%.
eq_var	Se **eq_var** for especificado como False (0), variâncias separadas das duas amostras serão assumidas. Se **eq_var** for especificado como True (1), variâncias iguais entre as duas amostras serão assumidas.

Limitações:

Valores de texto, valores NULL e valores ausentes nas expressões de valores farão com que a função retorne um resultado NULL.

Exemplos:

```
ZTest_lower( Group, Value )
ZTest_lower( Group, Value, sig, false )
```

Consulte também:

Exemplos de como usar funções z-test (page 534)

ZTest_upper

ZTest_upper() retorna o valor agregado da extremidade superior do intervalo de confiança, referente a duas séries independentes de valores.

Essa função se aplica a testes T de Student com amostras independentes.

Se a função for usada no script de carga dos dados, os valores serão iterados com base em um número de registros que são definidos por uma cláusula group by.

Se a função for usada para expressões de gráficos, os valores serão iterados com base nas dimensões do gráfico.

Sintaxe:

```
ZTest_upper (grp, value [, sig [, eq_var]])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
value	Os valores da amostra a serem avaliados. Os valores da amostra devem ser logicamente agrupados, como especificado por exatamente dois valores em **group**. Se um nome de campo dos valores das amostras não for fornecido no script de carregamento, o campo será automaticamente nomeado como **Value**.
grp	O campo que contém os nomes de cada um dos dois grupos de amostras. Se um nome de campo para o grupo não for fornecido no script de carregamento, o campo será automaticamente nomeado como **Type**.
sig	É possível especificar o nível de significância bicaudal em **sig**. Se omitido, **sig** será definido como 0,025, gerando um intervalo de confiança de 95%.
eq_var	Se **eq_var** for especificado como False (0), variâncias separadas das duas amostras serão assumidas. Se **eq_var** for especificado como True (1), variâncias iguais entre as duas amostras serão assumidas.

Limitações:

Valores de texto, valores NULL e valores ausentes nas expressões de valores farão com que a função retorne um resultado NULL.

Exemplos:

```
ZTest_upper( Group, Value )
ZTest_upper( Group, Value, sig, false )
```

Consulte também:

Exemplos de como usar funções z-test (page 534)

ZTestw_z

ZTestw_z() retorna o valor z agregado de duas séries de valores.

Essa função se aplica a testes z nos quais a série de dados de entrada é fornecida em um formato de duas colunas ponderadas.

Se a função for usada no script de carga dos dados, os valores serão iterados com base em um número de registros que são definidos por uma cláusula group by.

Se a função for usada para expressões de gráficos, os valores serão iterados com base nas dimensões do gráfico.

Sintaxe:

```
ZTestw_z (weight, value [, sigma])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
value	Os valores devem ser retornados por **value**. Pressupõe-se uma média amostral de 0. Para que o teste seja realizado com base em uma média diferente, subtraia esse valor dos valores de amostra.
weight	Cada valor em **value** pode ser calculado uma ou mais vezes, de acordo com um valor de ponderação correspondente em **weight**.
sigma	Se for conhecido, o desvio padrão pode ser indicado em **sigma**. Se **sigma** for omitido, o desvio padrão de amostra real será utilizado.

Limitações:

Valores de texto, valores NULL e valores ausentes nas expressões de valores farão com que a função retorne um resultado NULL.

Exemplo:

```
ZTestw_z( Weight, Value-TestValue)
```

Consulte também:

Exemplos de como usar funções z-test (page 534)

ZTestw_sig

ZTestw_sig() retorna o nível de significância agregado do teste z bicaudal, referente a uma série de valores.

Essa função se aplica a testes z nos quais a série de dados de entrada é fornecida em um formato de duas colunas ponderadas.

Se a função for usada no script de carga dos dados, os valores serão iterados com base em um número de registros que são definidos por uma cláusula group by.

Se a função for usada para expressões de gráficos, os valores serão iterados com base nas dimensões do gráfico.

Sintaxe:

```
ZTestw_sig (weight, value [, sigma])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
value	Os valores devem ser retornados por **value**. Pressupõe-se uma média amostral de 0. Para que o teste seja realizado com base em uma média diferente, subtraia esse valor dos valores de amostra.
weight	Cada valor em **value** pode ser calculado uma ou mais vezes, de acordo com um valor de ponderação correspondente em **weight**.
sigma	Se for conhecido, o desvio padrão pode ser indicado em **sigma**. Se **sigma** for omitido, o desvio padrão de amostra real será utilizado.

Limitações:

Valores de texto, valores NULL e valores ausentes nas expressões de valores farão com que a função retorne um resultado NULL.

Exemplo:

```
ZTestw_sig( Weight, Value-TestValue)
```

Consulte também:

Exemplos de como usar funções z-test (page 534)

ZTestw_dif

ZTestw_dif() retorna a diferença média agregada do teste z referente a uma série de valores.

Essa função se aplica a testes z nos quais a série de dados de entrada é fornecida em um formato de duas colunas ponderadas.

Se a função for usada no script de carga dos dados, os valores serão iterados com base em um número de registros que são definidos por uma cláusula group by.

Se a função for usada para expressões de gráficos, os valores serão iterados com base nas dimensões do gráfico.

Sintaxe:

```
ZTestw_dif ( weight, value [, sigma])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
value	Os valores devem ser retornados por **value**. Pressupõe-se uma média amostral de 0. Para que o teste seja realizado com base em uma média diferente, subtraia esse valor dos valores de amostra.
weight	Cada valor em **value** pode ser calculado uma ou mais vezes, de acordo com um valor de ponderação correspondente em **weight**.
sigma	Se for conhecido, o desvio padrão pode ser indicado em **sigma**. Se **sigma** for omitido, o desvio padrão de amostra real será utilizado.

Limitações:

Valores de texto, valores NULL e valores ausentes nas expressões de valores farão com que a função retorne um resultado NULL.

Exemplo:

```
ZTestw_dif( Weight, Value-TestValue)
```

Consulte também:

Exemplos de como usar funções z-test (page 534)

ZTestw_sterr

ZTestw_sterr() retorna o erro padrão agregado da diferença média do teste z, referente a uma série de valores.

Essa função se aplica a testes z nos quais a série de dados de entrada é fornecida em um formato de duas colunas ponderadas.

Se a função for usada no script de carga dos dados, os valores serão iterados com base em um número de registros que são definidos por uma cláusula group by.

Se a função for usada para expressões de gráficos, os valores serão iterados com base nas dimensões do gráfico.

Sintaxe:

```
ZTestw_sterr (weight, value [, sigma])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
value	Os valores devem ser retornados por **value**. Pressupõe-se uma média amostral de 0. Para que o teste seja realizado com base em uma média diferente, subtraia esse valor dos valores de amostra.
weight	Cada valor em **value** pode ser calculado uma ou mais vezes, de acordo com um valor de ponderação correspondente em **weight**.
sigma	Se for conhecido, o desvio padrão pode ser indicado em **sigma**. Se **sigma** for omitido, o desvio padrão de amostra real será utilizado.

Limitações:

Valores de texto, valores NULL e valores ausentes nas expressões de valores farão com que a função retorne um resultado NULL.

Exemplo:

```
ZTestw_sterr( Weight, Value-TestValue)
```

Consulte também:

Exemplos de como usar funções z-test (page 534)

ZTestw_conf

ZTestw_conf() retorna o valor de intervalo de confiança z agregado referente a uma série de valores.

Essa função se aplica a testes z nos quais a série de dados de entrada é fornecida em um formato de duas colunas ponderadas.

Se a função for usada no script de carga dos dados, os valores serão iterados com base em um número de registros que são definidos por uma cláusula group by.

Se a função for usada para expressões de gráficos, os valores serão iterados com base nas dimensões do gráfico.

Sintaxe:

```
ZTest_conf(weight, value[, sigma[, sig]])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
value	Os valores da amostra a serem avaliados. Pressupõe-se uma média amostral de 0. Para que o teste seja realizado com base em uma média diferente, subtraia essa média valor dos valores de amostra.
weight	Cada valor em **value** pode ser calculado uma ou mais vezes, de acordo com um valor de ponderação correspondente em **weight**.
sigma	Se for conhecido, o desvio padrão pode ser indicado em **sigma**. Se **sigma** for omitido, o desvio padrão de amostra real será utilizado.
sig	É possível especificar o nível de significância bicaudal em **sig**. Se omitido, **sig** será definido como 0,025, gerando um intervalo de confiança de 95%.

Limitações:

Valores de texto, valores NULL e valores ausentes nas expressões de valores farão com que a função retorne um resultado NULL.

Exemplo:

```
ZTestw_conf( Weight, Value-TestValue)
```

Consulte também:

Exemplos de como usar funções z-test (page 534)

ZTestw_lower

ZTestw_lower() retorna o valor agregado da extremidade inferior do intervalo de confiança, referente a duas séries independentes de valores.

Se a função for usada no script de carga dos dados, os valores serão iterados com base em um número de registros que são definidos por uma cláusula group by.

Se a função for usada para expressões de gráficos, os valores serão iterados com base nas dimensões do gráfico.

Sintaxe:

```
ZTestw_lower (grp, value [, sig [, eq_var]])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumento	Descrição
value	Os valores da amostra a serem avaliados. Os valores da amostra devem ser logicamente agrupados, como especificado por exatamente dois valores em **group**. Se um nome de campo dos valores das amostras não for fornecido no script de carregamento, o campo será automaticamente nomeado como **Value**.
grp	O campo que contém os nomes de cada um dos dois grupos de amostras. Se um nome de campo para o grupo não for fornecido no script de carregamento, o campo será automaticamente nomeado como **Type**.
sig	É possível especificar o nível de significância bicaudal em **sig**. Se omitido, **sig** será definido como 0,025, gerando um intervalo de confiança de 95%.
eq_var	Se **eq_var** for especificado como False (0), variâncias separadas das duas amostras serão assumidas. Se **eq_var** for especificado como True (1), variâncias iguais entre as duas amostras serão assumidas.

Limitações:

Valores de texto, valores NULL e valores ausentes nas expressões de valores farão com que a função retorne um resultado NULL.

Exemplos:

```
ZTestw_lower( Group, Value )
ZTestw_lower( Group, Value, sig, false )
```

Consulte também:

Exemplos de como usar funções z-test (page 534)

ZTestw_upper

ZTestw_upper() retorna o valor agregado da extremidade superior do intervalo de confiança, referente a duas séries independentes de valores.

Essa função se aplica a testes T de Student com amostras independentes.

Se a função for usada no script de carga dos dados, os valores serão iterados com base em um número de registros que são definidos por uma cláusula group by.

Se a função for usada para expressões de gráficos, os valores serão iterados com base nas dimensões do gráfico.

Sintaxe:

```
ZTestw_upper (grp, value [, sig [, eq_var]])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
value	Os valores da amostra a serem avaliados. Os valores da amostra devem ser logicamente agrupados, como especificado por exatamente dois valores em **group**. Se um nome de campo dos valores das amostras não for fornecido no script de carregamento, o campo será automaticamente nomeado como **Value**.
grp	O campo que contém os nomes de cada um dos dois grupos de amostras. Se um nome de campo para o grupo não for fornecido no script de carregamento, o campo será automaticamente nomeado como **Type**.
sig	É possível especificar o nível de significância bicaudal em **sig**. Se omitido, **sig** será definido como 0,025, gerando um intervalo de confiança de 95%.
eq_var	Se **eq_var** for especificado como False (0), variâncias separadas das duas amostras serão assumidas. Se **eq_var** for especificado como True (1), variâncias iguais entre as duas amostras serão assumidas.

Limitações:

Valores de texto, valores NULL e valores ausentes nas expressões de valores farão com que a função retorne um resultado NULL.

Exemplos:

```
ZTestw_upper( Group, Value )
ZTestw_upper( Group, Value, sig, false )
```

Consulte também:

Exemplos de como usar funções z-test (page 534)

Exemplo das funções de teste estatístico

Essa seção inclui exemplos de funções de teste estatístico como aplicado a gráficos e ao script de carregamento de dados.

Exemplos de como usar funções chi2-test em gráficos

As funções chi2-test são usadas para encontrar os valores associados com a análise estatística do Qui quadrado.

Esta seção descreve como criar visualizações usando dados de amostra para encontrar os valores das funções do teste de distribuição do Qui quadrado disponíveis no Qlik Sense. Consulte os tópicos individuais de função do gráficochi2-test para ver as descrições de sintaxe e argumentos.

Carregamento dos dados para as amostras

Há três conjuntos de dados de exemplo, descrevendo três amostras estatísticas diferentes a serem carregadas no script.

Faça o seguinte:

1. Crie um novo aplicativo.

   Na carga de dados, digite o seguinte:
   ```
   // 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. Clique em para carregar dados.

Criação das visualizações da função de gráfico chi2-test

Exemplo: Amostra 1

Faça o seguinte:

1. No editor de carregamento de dados, clique em para ir para a visualização de aplicativos e , em seguida, clique na pasta que você criou antes.
   A exibição da pasta é aberta.
2. Clique em **Editar pasta** para editar a pasta.
3. Em **Gráficos**, adicione uma tabela e em **Campos**, adicione Grp, Grade e Count como dimensões.
   Esta tabela contém os dados de amostra.
4. Adicione outra tabela com a seguinte expressão como uma dimensão.
   `ValueList('p','df','Chi2')`
   Isso utiliza a função de dimensões sintéticas para criar rótulos para as dimensões, com os nomes das três funções chi2-test.
5. Adicione a seguinte expressão à tabela como uma medida.

```
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)))
```

   Isso tem o efeito de colocar o valor resultante de cada função chi2-test na tabela ao lado de sua dimensão sintética associada.
6. Defina a **Formatação do número** da medida como **Número** e **3Números significativos**.

Na expressão para a medida, a seguinte expressão poderia ser usada: `Pick(Match(ValueList ('p','df','Chi2'),'p','df','Chi2'),Chi2Test_p(Grp,Grade,Count),Chi2Test_df (Grp,Grade,Count),Chi2Test_Chi2(Grp,Grade,Count))`

Resultado:

A tabela resultante para as funções chi2-test para os dados da Amostra 1 conterá os seguintes valores:

Tabela de resultados

p	df	Chi2
0.820	5	2.21

Exemplo: Amostra 2

Faça o seguinte:

1. Na pasta que você estava editando no exemplo Amostra 1, em **Gráficos**, adicione uma tabela e em **Campos**, adicione Sex, Opinion e OpCount como dimensões.
2. Faça uma cópia da tabela de resultados da Amostra 1, usando os comandos **Copiar** e **Colar**. Edite a expressão na medição e substitua os argumentos em todas as três funções chi2-test pelos nomes dos campos utilizados nos dados da Amostra 2, por exemplo: `Chi2Test_p(Sex,Opinion,OpCount)`.

Resultado:

A tabela resultante para as funções chi2-test para os dados da Amostra 2 conterá os seguintes valores:

Tabela de resultados

p	df	Chi2
0.000309	2	16.2

Exemplo: Amostra 3

Faça o seguinte:

1. Crie mais duas tabelas da mesma maneira que nos exemplos para os dados das Amostras 1 e 2. Na tabela de dimensões, use os seguintes campos como dimensões: Gender, Description, Actual e Expected.
2. Na tabela de resultados, use os nomes dos campos utilizados nos dados da Amostra 3, por exemplo: `Chi2Test_p(Gender,Description,Actual,Expected)`.

Resultado:

A tabela resultante para as funções chi2-test para os dados da Amostra 3 conterá os seguintes valores:

Tabela de resultados

p	df	Chi2
0.000308	2	16.2

Exemplos de como usar funções chi2-test no script de carregamento de dados

As funções chi2-test são usadas para encontrar os valores associados com a análise estatística do Qui quadrado. Esta seção descreve como usar as funções de teste de distribuição qui-quadrado disponíveis no Qlik Sense no script de carregamento de dados. Consulte os tópicos individuais de função de scriptchi2-test para as descrições de sintaxe e argumentos.

Este exemplo usa uma tabela contendo o número de estudantes obtendo uma nota (A-F) para dois grupos de estudantes (I e II).

Data table

Group	A	B	C	D	E	F
I	15	7	9	20	26	19
II	10	11	7	15	21	16

Carregamento dos dados de exemplo

Faça o seguinte:

1. Crie um novo aplicativo.

   No editor da carga de dados, digite o seguinte:
   ```
   // 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. Clique em para carregar dados.

Você carregou os novos dados de amostra.
   ```

Carregando os valores de função chi2-test

Ninguém carregará os valores chi2-test com base nos dados de amostra em uma nova tabela, agrupada por Grp.

Faça o seguinte:

1. No editor de carregamento de dados, adicione o seguinte no final do script:
```
// 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. Clique em para carregar dados.

Agora, você carregou os valores chi2-test em uma tabela chamada Chi2_table.

Resultados

Você pode visualizar os valores resultantes chi2-test no visualizador de modelo de dados em **Visualizar**, eles devem ficar assim:

Results

Grp	chi2	df	p
I	16.00	5	0.007
II	9.40	5	0.094

Criando um relatório típico de t-test

Um relatório de Student típico de t-test pode incluir tabelas com **Group Statistics** e resultados **Independent Samples Test**.

Nas próximas seções, criaremos essas tabelas usando funções do Qlik Senset-test aplicadas a dois grupos independentes de amostras, Observation e Comparison. As tabelas correspondentes para essas amostras ficariam assim:

Estatísticas de grupos

Type	N	Mean	Standard Deviation	Standard Error Mean
Comparison	20	11.95	14.61245	3.2674431
Observation	20	27.15	12.507997	2.7968933

Teste de amostra independente

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

Carregamento dos dados de exemplo

Faça o seguinte:

1. Crie um novo aplicativo com uma nova planilha.
2. Digite o seguinte no editor da carga de dados:
```
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 '|');
```
Neste script de carregamento, **recno()** é incluído porque **crosstable** exige três argumentos. Então, **recno()** simplesmente fornece um argumento extra, neste caso um ID para cada linha. Sem ele, os valores de amostra **Comparison** não seriam carregados.
3. Clique em para carregar dados.

Criação da tabela Group statistics

Faça o seguinte:

1. No editor da carga de dados, clique em para ir para visualização de aplicativos e , em seguida, clique na pasta que você criou antes.
   Isso abre a exibição da pasta.
2. Clique em **Editar pasta** para editar a pasta.
3. Em **Gráficos**, adicione uma tabela e, em **Campos**, adicione Type como uma dimensão à tabela.
4. Adicione as seguintes expressões como medidas.

   Exemplos de expressões

Rótulo	Expressão
N	Count(Value)
Mean	Avg(Value)
Standard Deviation	Stdev(Value)
Standard Error Mean	Sterr(Value)

5. Clique em **Classificação** e certifique-se de que Type está no topo da lista de classificação.

Resultado:

A tabela Group statistics para essas amostras ficariam assim:

Estatísticas de grupos

Type	N	Mean	Standard Deviation	Standard Error Mean
Comparison	20	11.95	14.61245	3.2674431
Observation	20	27.15	12.507997	2.7968933

Criação da tabela Independent sample test

Faça o seguinte:

1. Clique em **Editar pasta** para editar a pasta.
2. Em **Gráficos**, adicione uma tabela com a seguinte expressão como uma dimensão à tabela. `=ValueList (Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1))` e dê a ele o rótulo Tipo.

3. Adicione as seguintes expressões como medidas:

Exemplos de expressões

Rótulo	Expressão
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))

Resultado:

Teste de amostra independente

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

Exemplos de como usar funções z-test

As funções z-test são usadas para encontrar valores associados à análise estatística z-test para amostras de dados grandes, geralmente superiores a 30, e onde a variância é conhecida.

Esta seção descreve como criar visualizações usando dados de amostra para encontrar os valores das funções z-test disponíveis no Qlik Sense. Consulte os tópicos individuais de função do gráficoz-test para ver as descrições de sintaxe e argumentos.

Carregamento dos dados de exemplo

Os dados de amostra utilizados aqui são os mesmos usados nos exemplos da função t-test. O tamanho da amostra de dados seria normalmente considerado muito pequeno para a análise do teste Z, mas ele é suficiente para o objetivo de ilustrar a utilização das diferentes funções z-test no Qlik Sense.

Faça o seguinte:

1. Crie um novo aplicativo com uma nova planilha.

   Se você criou um aplicativo para as funções t-test, pode usá-lo e criar uma nova pasta para essas funções.

2. No editor da carga de dados, digite o seguinte:

```
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 '|');
```

   Neste script de carregamento, **recno()** é incluído porque **crosstable** exige três argumentos. Então, **recno()** simplesmente fornece um argumento extra, neste caso um ID para cada linha. Sem ele, os valores de amostra **Comparison** não seriam carregados.

3. Clique em para carregar dados.

Criação da tabela z-test

Faça o seguinte:

1. No editor de carregamento de dados, clique em para ir até a visualização do aplicativo e, em seguida, clique na pasta que você criou acima.
   A exibição da pasta é aberta.
2. Clique em **Editar pasta** para editar a pasta.
3. Em **Gráficos**, adicione uma tabela e em **Campos**, adicione Type como uma dimensão.
4. Adicione as seguintes expressões à tabela como medidas

Exemplos de expressões

Rótulo	Expressão
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)

*Você pode querer ajustar a formatação numérica das medidas para ver valores significativos. A tabela será mais fácil de ser lida se a formatação de números for definida para a maioria das medidas para **Number>Simple**, em vez de **Auto**. Mas para ZTest Sig, por exemplo, use a formatação de números: **Personalizado** e depois ajuste o padrão de formato para **#.######**.*

Resultado:

A tabela resultante para as funções z-test dos dados da amostra conterá os seguintes valores:

Tabela de resultados z-test

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

Criação da tabela z-testw

As funções z-testw devem ser usadas quando a série de dados de entrada ocorre em formato de duas colunas ponderadas. As expressões exigem um valor para o argumento weight.

Os exemplos descritos aqui usam sempre o valor 2, mas uma expressão poderia ser usada, que definiria um valor para weight para cada observação.

Faça o seguinte:

1. No editor de carregamento de dados, clique em para ir até a visualização do aplicativo e, em seguida, clique na pasta que você criou acima.
   A exibição da pasta é aberta.
2. Clique em **Editar pasta** para editar a pasta.
3. Em **Gráficos**, adicione uma tabela e em **Campos**, adicione Type como uma dimensão.
4. Adicione as seguintes expressões à tabela como medidas.

   Exemplos de expressões

Rótulo	Expressão
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)

   Use a mesma formatação de números do exemplo de funções z-test.

Resultado:

A tabela resultante para as funções z-testw conterão os seguintes valores:

Tabela de resultados z-testw

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

Funções de agregação de string

Esta seção descreve as funções de agregação relacionadas a cadeias de caracteres.

Cada função é descrita adicionalmente após a visão geral. Você também pode clicar no nome da função na sintaxe para acessar imediatamente os detalhes dessa função específica.

Funções de agregação de cadeias de caracteres no script de carga de dados

Concat

Concat() é usado para combinar valores dos caracteres. A função de script retorna a concatenação da string agregada de todos os valores da expressão iterados sobre um número de registros que são definidos por uma cláusula **group by**.

```
Concat ([ distinct ] expression [, delimiter [, sort-weight]])
```

FirstValue

FirstValue() retorna o valor que foi carregado pela primeira vez a partir dos registros definidos pela expressão, classificado por uma cláusula **group by**.

Essa função só está disponível como função de script.

```
FirstValue (expression)
```

LastValue

LastValue() retorna o valor que foi carregado pela última vez a partir dos registros definidos pela expressão, classificado por uma cláusula **group by**.

Essa função só está disponível como função de script.

```
LastValue (expression)
```

MaxString

MaxString() encontra valores na expressão e retorna o último valor de texto classificado em um número de registros que são definidos por uma cláusula **group by**.

```
MaxString (expression )
```

MinString

MinString() encontra valores na expressão e retorna o primeiro valor de texto classificado em um número de registros que são definidos por uma cláusula **group by**.

```
MinString (expression )
```

Funções de agregação de caracteres em gráficos

As seguintes funções de gráfico estão disponíveis para agregar strings em gráficos.

Concat

Concat() é usado para combinar valores dos caracteres. A função retorna a concatenação de caracteres agregada de todos os valores da expressão avaliada em cada dimensão.

```
Concat - função de gráfico({[SetExpression] [DISTINCT] [TOTAL [<fld{, fld}>]]
string[, delimiter[, sort_weight]]])
```

MaxString

MaxString() encontra valores de caracteres na expressão ou campo e retorna o último valor de texto na ordem de classificação alfabética.

```
MaxString - função de gráfico({[SetExpression] [TOTAL [<fld{, fld}>]]} expr)
```

MinString

MinString() encontra valores de caracteres na expressão ou campo e retorna o primeiro valor de texto na ordem de classificação alfabética.

```
MinString - função de gráfico({[SetExpression] [TOTAL [<fld {, fld}>]]} expr)
```

Concat

Concat() é usado para combinar valores dos caracteres. A função de script retorna a concatenação da string agregada de todos os valores da expressão iterados sobre um número de registros que são definidos por uma cláusula **group by**.

Sintaxe:

```
Concat ([ distinct ] string [, delimiter [, sort-weight]])
```

Tipo de dados de retorno: caractere

Argumentos:

A expressão ou campo que contém a string a ser processada.

Argumentos

Argumento	Descrição
string	A expressão ou campo que contém a string a ser processada.
delimiter	Cada valor pode ser separado pelo caractere encontrado em delimiter.
sort-weight	A ordem de concatenação pode ser determinada pelo valor de dimensão **sort-weight**, caso esteja presente, com o caractere correspondente ao valor mais baixo aparecendo primeiro na concatenação.
distinct	Se a palavra **distinct** aparecer antes da expressão, todas as duplicatas serão ignoradas.

Exemplos e resultados:

Adicione o script de exemplo ao seu aplicativo e execute-o. Para ver o resultado, adicione os campos listados na coluna de resultados a uma pasta no seu aplicativo.

Exemplos e resultados

Exemplo	Resultado	Resultados depois de adicionados a uma pasta

`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

Dado que a tabela **TeamData** é carregada como no exemplo anterior: `LOAD SalesGroup,Concat(distinct Team,'-') as TeamConcat2 Resident TeamData Group By SalesGroup;`	SalesGroup East West	TeamConcat2 Alpha-Beta-Delta-Gamma Epsilon-Eta-Theta-Zeta
Considerando que a tabela **TeamData** esteja carregada, como no exemplo anterior. Como o argumento para **sort-weight** é adicionado, os resultados são ordenados pelo valor da dimensão Amount: `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 - função de gráfico

Concat() é usado para combinar valores dos caracteres. A função retorna a concatenação de caracteres agregada de todos os valores da expressão avaliada em cada dimensão.

Sintaxe:

```
Concat({[SetExpression] [DISTINCT] [TOTAL [<fld{, fld}>]]} string[, delimiter [, sort_weight]])
```

Tipo de dados de retorno: caractere

Argumentos:

Argumentos

Argumento	Descrição
string	A expressão ou campo que contém a string a ser processada.
delimiter	Cada valor pode ser separado pelo caractere encontrado em delimiter.
sort-weight	A ordem de concatenação pode ser determinada pelo valor de dimensão **sort-weight**, caso esteja presente, com o caractere correspondente ao valor mais baixo aparecendo primeiro na concatenação.
SetExpression	Por padrão, a função de agregação agregará um conjunto de registros possíveis definidos pela seleção. Um conjunto de registros alternativos pode ser definido por uma expressão de análise.
DISTINCT	Se a palavra **DISTINCT** aparecer antes dos argumentos de função, as duplicatas resultantes da avaliação dos argumentos de função serão ignoradas.
TOTAL	Se a palavra **TOTAL** ocorrer antes dos argumentos da função, o cálculo será feito sobre todos os valores possíveis, dadas as seleções atuais, e não apenas sobre os pertinentes ao valor dimensional atual, isto é, desconsiderando as dimensões do gráfico. Usando **TOTAL [<fld {.fld}>]**, em que o qualificador **TOTAL** é seguido por uma lista de um ou mais nomes de campo como um subconjunto das variáveis de dimensão de gráfico, você cria um subconjunto dos valores possíveis totais.

Exemplos e resultados:

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

Exemplos de funções

Exemplo	Resultado
`Concat(Team)`	A tabela é criada a partir das dimensões SalesGroup e Amount e das variações da medida Concat(Team). Ignorando o resultado total, observe que embora existam dados para oito valores de Team distribuídos para dois valores de SalesGroup, o único resultado da medida Concat(Team) que concatena mais de um valor de caracteres Team na tabela é a linha que contém a dimensão Amount 20000, que dá o resultado BetaGammaGamma. Isso ocorre porque existem três valores para Amount 20000 nos dados de entrada. Todos os demais resultados permanecem desconcatenados quando a medida é estendida para as dimensões, porque há apenas um valor de Team para cada combinação de SalesGroup e Amount.
`Concat (DISTINCT Team,', ')`	Beta, Gamma. porque o qualificador DISTINCT significa que o resultado Gamma duplicado é desconsiderado. Além disso, o argumento do delimitador é definido como uma vírgula seguida por um espaço.
`Concat (TOTAL <SalesGroup> Team)`	Todos os valores de string para todos os valores de Team são concatenados se o qualificador TOTAL for usado. Com a seleção de campo <SalesGroup> especificada, isso divide os resultados em dois valores da dimensão SalesGroup. Para o SalesGroupEast, os resultados são AlphaBetaDeltaGammaGamma. Para o SalesGroupWest, os resultados são EpsilonEtaThetaZeta.
`Concat (TOTAL <SalesGroup> Team,';', Amount)`	Ao adicionar o argumento para **sort-weight**: Amount, os resultados são ordenados pelo valor da dimensão Amount. Os resultados tornam-se DeltaBetaGammaGammaAlpha e EtaEpsilonZEtaTheta.

Dados usados no exemplo:

```
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() retorna o valor que foi carregado pela primeira vez a partir dos registros definidos pela expressão, classificado por uma cláusula **group by**.

Essa função só está disponível como função de script.

Sintaxe:

```
FirstValue ( expr)
```

Tipo de dados de retorno: dual

Argumentos:

Argumentos

Argumento	Descrição
expr	A expressão ou campo que contém os dados a serem medidos.

Limitações:

Se nenhum valor de texto for encontrado, retornará NULL.

Exemplos e resultados:

Adicione o script de exemplo ao seu aplicativo e execute-o. Para ver o resultado, adicione os campos listados na coluna de resultados a uma pasta no seu aplicativo.

Dados resultantes

Exemplo	Resultado	Resultados em uma pasta

`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() retorna o valor que foi carregado pela última vez a partir dos registros definidos pela expressão, classificado por uma cláusula **group by**.

Essa função só está disponível como função de script.

Sintaxe:

```
LastValue ( expr )
```

Tipo de dados de retorno: dual

Argumentos:

Argumentos

Argumento	Descrição
expr	A expressão ou campo que contém os dados a serem medidos.

Limitações:

Se nenhum valor de texto for encontrado, retornará NULL.

Exemplos e resultados:

Adicione o script de exemplo ao seu aplicativo e execute-o. Em seguida, adicione pelo menos os campos listados na coluna de resultados a uma pasta para ver o resultado.

Para obter a mesma aparência que na coluna de resultados abaixo, no painel de propriedades, em Classificação, alterne de Automática para Personalizada e desmarque a classificação numérica e alfabética.

Exemplo	Resultado	Resultado com ordenação personalizada
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() encontra valores na expressão e retorna o último valor de texto classificado em um número de registros que são definidos por uma cláusula **group by**.

Sintaxe:

```
MaxString ( expr )
```

Tipo de dados de retorno: dual

Argumentos:

Argumento	Descrição
expr	A expressão ou campo que contém os dados a serem medidos.

Limitações:

Se nenhum valor de texto for encontrado, retornará NULL.

Exemplos e resultados:

Adicione o script de exemplo ao seu aplicativo e execute-o. Para ver o resultado, adicione os campos listados na coluna de resultados a uma pasta no seu aplicativo.

Exemplo	Resultado	
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
Considerando que a tabela **TeamData** está carregada como no exemplo anterior, e seu script de carregamento de dados tem a instrução SET: 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 - função de gráfico

MaxString() encontra valores de caracteres na expressão ou campo e retorna o último valor de texto na ordem de classificação alfabética.

Sintaxe:

```
MaxString({[SetExpression] [TOTAL [<fld{, fld}>]]} expr)
```

Tipo de dados de retorno: dual

Argumentos:

Argumentos

Argumento	Descrição
expr	A expressão ou campo que contém os dados a serem medidos.
SetExpression	Por padrão, a função de agregação agregará um conjunto de registros possíveis definidos pela seleção. Um conjunto de registros alternativos pode ser definido por uma expressão de análise.
TOTAL	Se a palavra **TOTAL** ocorrer antes dos argumentos da função, o cálculo será feito sobre todos os valores possíveis, dadas as seleções atuais, e não apenas sobre os pertinentes ao valor dimensional atual, isto é, desconsiderando as dimensões do gráfico. Usando **TOTAL [<fld {.fld}>]**, em que o qualificador **TOTAL** é seguido por uma lista de um ou mais nomes de campo como um subconjunto das variáveis de dimensão de gráfico, você cria um subconjunto dos valores possíveis totais.

Limitações:

Se a expressão não contiver valores com uma representação de string, NULL é retornado.

Exemplos e resultados:

Tabela de resultados

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

Exemplos de funções

Exemplo	Resultado
`MaxString (Team)`	Há três valores de 20000 para a dimensão Amount: dois de Gamma (em datas diferentes), e um de Beta. Portanto, o resultado da medição MaxString (Team) é Gamma, porque esse é o mais alto valor nas strings ordenadas.
`MaxString (Date)`	2013/11/01 é o maior valor Date entre os três associados à dimensão Amount. Isso pressupõe que o seu script tem o comando SET `SET DateFormat='YYYY-MM-DD';'`

Dados usados no exemplo:

```
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() encontra valores na expressão e retorna o primeiro valor de texto classificado em um número de registros que são definidos por uma cláusula **group by**.

Sintaxe:

```
MinString ( expr )
```

Tipo de dados de retorno: dual

Argumentos:

Argumentos

Argumento	Descrição
expr	A expressão ou campo que contém os dados a serem medidos.

Limitações:

Se nenhum valor de texto for encontrado, retornará NULL.

Exemplos e resultados:

Adicione o script de exemplo ao seu aplicativo e execute-o. Para ver o resultado, adicione os campos listados na coluna de resultados a uma pasta no seu aplicativo.

Dados resultantes

Exemplo	Resultado	
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
Considerando que a tabela **TeamData** está carregada como no exemplo anterior, e seu script de carregamento de dados tem a instrução SET: 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 - função de gráfico

MinString() encontra valores de caracteres na expressão ou campo e retorna o primeiro valor de texto na ordem de classificação alfabética.

Sintaxe:

```
MinString({[SetExpression] [TOTAL [<fld {, fld}>]]} expr)
```

Tipo de dados de retorno: dual

Argumentos:

Argumentos

Argumento	Descrição
expr	A expressão ou campo que contém os dados a serem medidos.
SetExpression	Por padrão, a função de agregação agregará um conjunto de registros possíveis definidos pela seleção. Um conjunto de registros alternativos pode ser definido por uma expressão de análise.

Argumento	Descrição
TOTAL	Se a palavra **TOTAL** ocorrer antes dos argumentos da função, o cálculo será feito sobre todos os valores possíveis, dadas as seleções atuais, e não apenas sobre os pertinentes ao valor dimensional atual, isto é, desconsiderando as dimensões do gráfico. Usando **TOTAL [<fld {.fld}>]**, em que o qualificador **TOTAL** é seguido por uma lista de um ou mais nomes de campo como um subconjunto das variáveis de dimensão de gráfico, você cria um subconjunto dos valores possíveis totais.

Exemplos e resultados:

Dados de exemplo

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

Exemplos de funções

Exemplos	Resultados
`MinString (Team)`	Há três valores de 20000 para a dimensão Amount: dois de Gamma (em datas diferentes), e um de Beta. Portanto, o resultado da medição MinString (Team) é Beta, porque esse é o mais alto valor nas strings ordenadas.
`MinString (Date)`	2013/11/01 é o primeiro valor Date entre os três associados à dimensão Amount. Isso pressupõe que o seu script tem o comando SET `SET DateFormat='YYYY-MM-DD';'`

Dados usados no exemplo:

```
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 '|');
```

Funções de dimensão sintética

A dimensão sintética é criada no aplicativo a partir dos valores gerados das funções de dimensão sintética, e não diretamente de campos no modelo de dados. Quando os valores gerados por uma função de dimensão sintética são utilizados em um gráfico como uma dimensão calculada, isso cria uma dimensão sintética. As dimensões sintéticas permitem criar, por exemplo, gráficos de dimensões com valores decorrentes de seus dados, isto é, dimensões dinâmicas.

Dimensões sintéticas não são afetadas pelas seleções.

As seguintes funções de dimensão sintética podem ser usadas em gráficos.

ValueList

ValueList() retorna um conjunto de valores listados que, quando usados em uma dimensão calculada, formarão uma dimensão sintética.

```
ValueList - função de gráfico (v1 {, Expression})
```

ValueLoop

ValueLoop() retorna um conjunto de valores repetidos que, quando usados em uma dimensão calculada, formarão uma dimensão sintética.

```
ValueLoop - função de gráfico(from [, to [, step ]])
```

ValueList - função de gráfico

ValueList() retorna um conjunto de valores listados que, quando usados em uma dimensão calculada, formarão uma dimensão sintética.

*Em gráficos com uma dimensão sintética criada com a função **ValueList** é possível fazer referência ao valor da dimensão correspondente a uma célula de expressão específica declarando novamente a função **ValueList** com os mesmos parâmetros na expressão de gráfico. A função pode, é claro, ser usada em qualquer lugar do layout. No entanto, exceto quando usada para dimensões sintéticas, ela só terá significado dentro de uma função de agregação.*

Dimensões sintéticas não são afetadas pelas seleções.

Sintaxe:

```
ValueList(v1 {,...})
```

Tipo de dados de retorno: dual

Argumentos:

Argumentos

Argumento	Descrição
v1	Valor estático (normalmente uma string, mas pode ser um número).
{,...}	Lista opcional de valores estáticos.

Exemplos e resultados:

Exemplos de funções

Exemplo	Resultado
`ValueList ('Number of Orders', 'Average Order Size', 'Total Amount')`	Quando usado para criar uma dimensão em uma tabela, por exemplo, isso resulta em três valores de string como rótulos de linha na tabela. Eles podem ser então referenciados em uma expressão.
`=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) ))`	Essa expressão obtém os valores da dimensão criada e os referencia em uma instrução IF aninhada como entrada para três funções de agregação: **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

Dados usados nos exemplos:

```
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 - função de gráfico

ValueLoop() retorna um conjunto de valores repetidos que, quando usados em uma dimensão calculada, formarão uma dimensão sintética.
Os valores gerados começarão com o valor **from** e terminarão com o valor **to** incluindo valores intermediários em incrementos de passo.

*Em gráficos com uma dimensão sintética criada com a função **ValueLoop** é possível fazer referência ao valor da dimensão correspondente a uma célula de expressão específica declarando novamente a função **ValueLoop** com os mesmos parâmetros na expressão de gráfico. A função pode, é claro, ser usada em qualquer lugar do layout. No entanto, exceto quando usada para dimensões sintéticas, ela só terá significado dentro de uma função de agregação.*

Dimensões sintéticas não são afetadas pelas seleções.

Sintaxe:

```
ValueLoop(from [, to [, step ]])
```

Tipo de dados de retorno: dual

Argumentos:

Argumentos

Argumentos	Descrição
from	Valor de início no conjunto de valores que serão gerados.
to	Valor de término no conjunto de valores que serão gerados.
step	Tamanho do incremento entre os valores.

Exemplos e resultados:

Exemplos de funções

Exemplo	Resultado
`ValueLoop (1, 10)`	Isso cria uma dimensão de uma tabela, por exemplo, que pode ser usado para objetivos como a rotulagem numerada. Os exemplos descritos aqui resultam em valores numerados de 1 a 10. Estes valores podem ser então referenciados em uma expressão.
`ValueLoop (2, 10,2)`	Esse exemplo resulta em valores numerados como 2, 4, 6, 8 e 10 porque o argumento step tem um valor de 2.

Agregações aninhadas

Você pode se deparar com situações em que precisa aplicar uma agregação ao resultado de outra agregação. Isso é chamado de agregações aninhadas.

Não é possível aninhar agregações na maioria das expressões de gráfico. Porém, você poderá aninhar agregações se usar o qualificador **TOTAL** na função de agregação interna.

São permitidos no máximo 100 níveis de aninhamento.

Agregações aninhadas com o qualificador TOTAL

Exemplo:

Você deseja calcular a soma do campo **Sales**, mas só quer incluir as transações com **OrderDate** igual ao último ano. O último ano pode ser obtido com a função de agregação **`Max(TOTAL Year(OrderDate))`**.

A seguinte agregação retornaria o resultado desejado:

```
Sum(If(Year(OrderDate)=Max(TOTAL Year(OrderDate)), Sales))
```

O Qlik Sense exige a inclusão do qualificador **TOTAL** nesse tipo de aninhamento. Isso é necessário para a comparação desejada. Esse tipo de necessidade de aninhamento é bastante comum e é uma boa prática.

Consulte também:

Aggr - função de gráfico (page 552)

5.3 Aggr - função de gráfico

Aggr() retorna um conjunto de valores da expressão calculada sobre a dimensão indicada e as dimensões. Por exemplo, o valor máximo de vendas por cliente, por região.

A função **Aggr** é utilizada para agregações aninhadas, nas quais o primeiro parâmetro (a agregação interna) é calculado uma vez por valor dimensional. As dimensões são especificadas no segundo parâmetro (e parâmetros subsequentes).

Além disso, a função **Aggr** deve ser incluída em uma função de agregação externa usando a matriz de resultados da função **Aggr** como entrada para a agregação na qual ela está aninhada.

Sintaxe:

```
Aggr({SetExpression}[DISTINCT] [NODISTINCT ] expr, StructuredParameter{,
StructuredParameter})
```

Tipo de dados de retorno: dual

Argumentos:

Argumentos

Argumento	Descrição
expr	Uma expressão que consiste em uma função de agregação. Por padrão, a função de agregação agregará um conjunto de registros possíveis definidos pela seleção.
StructuredParameter	StructuredParameter consiste em uma dimensão e, opcionalmente, em critérios de classificação no seguinte formato: `(Dimension(Sort-type, Ordering))` A dimensão é um campo único e não pode ser uma expressão. A dimensão é usada para determinar a matriz de valores para a qual a expressão Aggr é calculada. Se critérios de classificação estiverem incluídos, a matriz de valores criada pela função Aggr, calculada para a dimensão, será classificada. Isso é importante quando a ordem de classificação afeta o resultado da expressão da função Aggr em que está delimitada. Para obter detalhes de como usar critérios de classificação, consulte Adicionando critérios de classificação à dimensão no parâmetro estruturado.
SetExpression	Por padrão, a função de agregação agregará um conjunto de registros possíveis definidos pela seleção. Um conjunto de registros alternativos pode ser definido por uma expressão de análise.
DISTINCT	Se o argumento da expressão for precedido pelo qualificador **distinct** ou se não for usado nenhum qualificador, cada combinação distinta de valores de dimensão gerará somente um valor de retorno. Esta é a maneira como as agregações normais são criadas: cada combinação distinta de valores de dimensão gera uma linha no gráfico.
NODISTINCT	Se o argumento da expressão for precedido pelo qualificador **nodistinct**, cada combinação de valores de dimensão poderá gerar mais de um valor de retorno, dependendo da estrutura de dados subjacente. Se houver apenas uma dimensão, a função **aggr** retornará uma matriz com o mesmo número de elementos que o número de linhas nos dados de origem.

Funções de agregação básicas, como **Sum**, **Min** e **Avg**, retornam um único valor numérico, enquanto a função Aggr() pode ser comparada à criação de um conjunto de resultados temporários em etapas (uma tabela virtual) com os quais outra agregação pode ser feita. Por exemplo, calculando uma média do valor de vendas ao somar as vendas por cliente em um comando **Aggr()** e depois calculando a média dos resultados somados: **Avg(TOTAL Aggr(Sum(Sales),Customer))**.

Use a função Aggr() em dimensões calculadas se quiser criar agregações de gráfico aninhadas em vários níveis.

Limitações:

Cada dimensão em uma função Aggr() deve ser um campo único e não pode ser uma expressão (dimensão calculada).

Adicionando critérios de classificação à dimensão no parâmetro estruturado

Em sua forma básica, o argumento StructuredParameter na sintaxe da função Aggr é uma dimensão única. A expressão: Aggr(Sum(Sales, Month)) encontra o valor total das vendas para cada mês. No entanto, quando delimitada em outra função de agregação, pode haver resultados inesperados, a menos que sejam utilizados critérios de classificação. Isso porque algumas dimensões podem ser classificadas por ordem alfabética ou numérica, e assim por diante.

No argumento StructuredParameter da função Aggr, você pode especificar critérios de classificação na dimensão em sua expressão. Dessa forma, você impõe uma ordem de classificação na tabela virtual que é produzida pela função Aggr.

O argumento StructuredParameter tem a seguinte sintaxe:

```
(FieldName, (Sort-type, Ordering))
```

Parâmetros estruturados podem ser aninhados:

```
(FieldName, (FieldName2, (Sort-type, Ordering)))
```

O tipo de classificação pode ser: NUMERIC, TEXT, FREQUENCY ou LOAD_ORDER.

Os tipos de ordenação associados a cada tipo de classificação são os seguintes:

Tipos de ordenação permitidos

Tipo de classificação	**Tipos de ordenação permitidos**
NUMERIC	ASCENDING, DESCENDING ou REVERSE
TEXT	ASCENDING, A2Z, DESCENDING, REVERSE ou Z2A
FREQUENCY	DESCENDING, REVERSE ou ASCENDING
LOAD_ORDER	ASCENDING, ORIGINAL, DESCENDING ou REVERSE

Os tipos de ordenação REVERSE e DESCENDING são equivalentes.

Para o Tipo de classificação TEXT, os tipos de ordenação ASCENDING e A2Z são equivalentes, e DESCENDING, REVERSE e Z2A são equivalentes.

Para o Tipo de classificação LOAD_ORDER, os tipos de ordenação ASCENDING e ORIGINAL são equivalentes.

Exemplos: Expressões de gráfico usando Agg

Exemplos - expressões de gráfico

Exemplo 1 de expressão de gráfico

Script de carregamento

Carregue os seguintes dados como um carregamento inline no editor de carregamento de dados para criar o exemplo de expressão de gráfico abaixo.

```
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 '|');
```

Expressão de gráfico

Crie uma visualização de KPI em uma pasta do Qlik Sense. Adicione a seguinte expressão ao KPI como uma medida:

```
Avg(Aggr(Sum(UnitSales*UnitPrice), Customer))
```

Resultado

376.7

Explicação

A expressão `Aggr(Sum(UnitSales*UnitPrice), Customer)` encontra o valor total das vendas por **Customer** e retorna uma matriz de valores: 295, 715 e 120 para os três valores **Customer**.

Efetivamente, criamos uma lista temporária de valores sem ter que criar uma tabela ou coluna explícita contendo esses valores.

Esses valores são usados como entrada para a função de **Avg()** para encontrar a média do valor de vendas, 376.7.

Exemplo 2 de expressão de gráfico

Script de carregamento

Carregue os seguintes dados como um carregamento inline no editor de carregamento de dados para criar o exemplo de expressão de gráfico abaixo.

```
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 '|');
```

Expressão de gráfico

Crie uma visualização de tabela em uma pasta do Qlik Sense com **Customer**, **Product**, **UnitPrice** e **UnitSales** como dimensões. Adicione a seguinte expressão à tabela como uma medida:

```
Aggr(NODISTINCT Max(UnitPrice), Customer, Product)
```

Resultado

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

Explicação

Uma matriz de valores: 16, 16, 15, 15, 12, 12, 22, 22, 25, 15, 15 e 19. O qualificador **nodistinct** significa que a matriz contém um elemento para cada linha na fonte de dados: cada um deles é o **UnitPrice** máximo para cada **Customer** e **Product**.

Exemplo 3 de expressão de gráfico

Script de carregamento

Carregue os seguintes dados como um carregamento inline no editor de carregamento de dados para criar o exemplo de expressão de gráfico abaixo.

```
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;
```

Expressões de gráfico

Crie uma visualização de tabela em uma pasta do Qlik Sense com **Year** e **Month** como dimensões. Adicione as seguintes expressões à tabela como medidas:

- `Sum(Sales)`
- `Sum(Aggr( Rangesum(Above(Sum(Sales),0,12)), (Year, (Numeric, Ascending)), (Month, (Numeric, Ascending)) ))` rotulado como Structured Aggr() na tabela.

Resultado

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

Explicação

Este exemplo exibe os valores agregados ao longo de um período de doze meses para cada ano em ordem cronológica crescente, daí a parte de parâmetros estruturados (Numeric, Ascending) da expressão **Aggr()**. Duas dimensões específicas são necessárias como parâmetros estruturados: **Year** e **Month**, classificados como (1) **Year** (numérico) e (2) **Month** (numérico). Essas duas dimensões devem ser usadas na visualização de tabela ou gráfico. Isso é necessário para que a lista de dimensões da função **Aggr()** corresponda às dimensões do objeto usado na visualização.

Você pode comparar a diferença entre essas medidas em uma tabela ou em gráficos de linhas separados:

- `Sum(Aggr( Rangesum(Above(Sum(Sales),0,12)), (Year), (Month) ))`
- `Sum(Aggr( Rangesum(Above(Sum(Sales),0,12)), (Year, (Numeric, Ascending)), (Month, (Numeric, Ascending)) ))`

Deve ficar claro que somente a última expressão executa o acúmulo desejado de valores agregados.

Consulte também:

Funções básicas de agregação (page 329)

5.4 Funções de cor

Estas funções podem ser usadas em expressões associadas à definição e avaliação das propriedades de cores dos objetos de gráfico, bem como em scripts de carga de dados.

*O Qlik Sense oferece suporte para as funções de cores **Color()**, **qliktechblue** e **qliktechgray** por motivos de compatibilidade com versões anteriores, mas sua utilização não é recomendada.*

ARGB

ARGB() é utilizada em expressões para definir ou avaliar as propriedades de cores de um objeto de gráfico, no qual a cor é definida por um componente vermelho **r**, um componente verde **g** e um componente azul **b**, com um fator alfa (opacidade) de **alpha**.

```
ARGB (alpha, r, g, b)
```

HSL

HSL() é utilizada em expressões para definir ou avaliar as propriedades de cores de um objeto gráfico, em que a cor é definida pelos valores de **hue**, **saturation** e **luminosity** entre 0 e 1.

```
HSL  (hue, saturation, luminosity)
```

RGB

RGB() retorna um número inteiro correspondente ao código da cor definida pelos três parâmetros: o componente vermelho R, o componente verde G e o componente azul B. Esses componentes devem ter valores inteiros entre 0 e 255. A função pode ser usada em expressões para definir ou avaliar as propriedades de cor de um objeto gráfico.

```
RGB  (r, g, b)
```

Colormix1

Colormix1() é usado em expressões para retornar uma representação da cor ARGB a partir de um gradiente de duas cores, com base em um valor entre 0 e 1.

```
Colormix1 (Value , ColorZero , ColorOne)
```

Value é um número real entre 0 e 1.

- Se Value = 0 ColorZero será retornado.
- Se Value = 1 ColorOne será retornado.
- Se o Value estiver entre 0 e 1, será retornado o sombreamento intermediário apropriado.

ColorZero é uma representação da cor RGB válida para a cor que será associada ao limite inferior do intervalo.

ColorOne é uma representação da cor RGB válida para a cor que será associada ao limite superior do intervalo.

Exemplo:

```
Colormix1(0.5, red(), blue())
```

retorna:

```
ARGB(255,64,0,64) (purple)
```

Colormix2

Colormix2() é usado em expressões para retornar uma representação da cor ARGB a partir de um gradiente de duas cores com base em um valor entre -1 e 1, com a possibilidade de especificar uma cor intermediária para a posição central (0).

```
Colormix2 (Value ,ColorMinusOne , ColorOne[ , ColorZero])
```

Value é um número real entre -1 e 1.

- Se Value = -1, a primeira cor será retornada.
- Se Value = 1, a primeira cor será retornada.
- Se -1 < Value< 1 a mistura apropriada de cores é retornada.

ColorMinusOne é uma representação da cor RGB válida para a cor que será associada ao limite inferior do intervalo.

ColorOne é uma representação da cor RGB válida para a cor que será associada ao limite superior do intervalo.

ColorZero é uma representação da cor RGB válida opcional para a cor que será associada ao valor central do intervalo.

SysColor

SysColor() retorna a representação da cor ARGB para a cor do sistema Windows nr, em que nr corresponde ao parâmetro da função da API do Windows **GetSysColor(nr)**.

```
SysColor (nr)
```

ColorMapHue

ColorMapHue() retorna um valor de cor ARGB do mapa de cores que varia o componente da tonalidade do modelo de cores HSV. O mapa de cores começa com vermelho, passa pelo amarelo, verde, ciano, azul, magenta e retorna ao vermelho. x deve ser especificado como um valor entre 0 e 1.

```
ColorMapHue (x)
```

ColorMapJet

ColorMapJet() retorna um valor de cor ARGB do mapa de cores que começa com azul, passa pelo ciano, amarelo, laranja e retorna ao vermelho. x deve ser especificado como um valor entre 0 e 1.

```
ColorMapJet (x)
```

Funções de cores pré-definidas

As funções a seguir podem ser usadas em expressões para cores pré-definidas. Cada função retorna uma representação de cor RGB.

Opcionalmente um parâmetro para o fator alfa pode ser fornecido, neste caso uma representação de cor ARGB é retornada. Um fator alfa de 0 corresponde à transparência total, e um de 255 corresponde à opacidade total. Se um valor para alfa não for inserido, o valor adotado será 255.

Funções de cores pré-definidas

Função de cor	RGB Valor
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)

Exemplos e resultados:

Exemplos e resultados

Exemplos	Resultados
`Blue()`	RGB(0,0,128)
`Blue(128)`	ARGB(128,0,0,128)

ARGB

ARGB() é utilizada em expressões para definir ou avaliar as propriedades de cores de um objeto de gráfico, no qual a cor é definida por um componente vermelho **r**, um componente verde **g** e um componente azul **b**, com um fator alfa (opacidade) de **alpha**.

Sintaxe:

```
ARGB(alpha, r, g, b)
```

Tipo de dados de retorno: dual

Argumentos:

Argumentos

Argumento	Descrição
alpha	Valor de transparência na faixa de 0 - 255. 0 corresponde a transparência total e 255, a opacidade total.
r, g, b	Valores de componentes vermelho, verde e azul. O componente de cor 0 corresponde à ausência de contribuição, e 255 à contribuição total.

Todos os argumentos devem ser expressões que resolvem para inteiros no intervalo de 0 a 255.

Se o componente numérico for interpretado e formatado em notação hexadecimal, a visualização dos valores dos componentes de cor torna-se mais fácil. Por exemplo, verde-claro tem o número 4 278 255 360, que em notação hexadecimal é FF00FF00. As duas primeiras posições "FF" (255) representam o canal **alfa**. As próximas duas posições "00" representam a quantidade de **vermelho**, as próximas duas posições "FF" representam a quantidade de **verde** e as duas posições finais "00" representam a quantidade de **azul**.

RGB

RGB() retorna um número inteiro correspondente ao código da cor definida pelos três parâmetros: o componente vermelho R, o componente verde G e o componente azul B. Esses componentes devem ter valores inteiros entre 0 e 255. A função pode ser usada em expressões para definir ou avaliar as propriedades de cor de um objeto gráfico.

Sintaxe:

```
RGB (r, g, b)
```

Tipo de dados de retorno: dual

Argumentos:

Argumentos

Argumento	Descrição
r, g, b	Valores de componentes vermelho, verde e azul. O componente de cor 0 corresponde à ausência de contribuição, e 255 à contribuição total.

Todos os argumentos devem ser expressões que resolvem para inteiros no intervalo de 0 a 255.

Se o componente numérico for interpretado e formatado em notação hexadecimal, a visualização dos valores dos componentes de cor torna-se mais fácil. Por exemplo, verde-claro tem o número 4 278 255 360, que em notação hexadecimal é FF00FF00. As duas primeiras posições "FF" (255) representam o canal **alfa**. Nas funções **RGB** e **HSL**, isso é sempre 'FF' (opaco). As próximas duas posições "00" representam a quantidade de **vermelho**, as próximas duas posições "FF" representam a quantidade de **verde** e as duas posições finais "00" representam a quantidade de **azul**.

Exemplo: Expressão de gráfico
Este exemplo aplica uma cor personalizada a um gráfico:

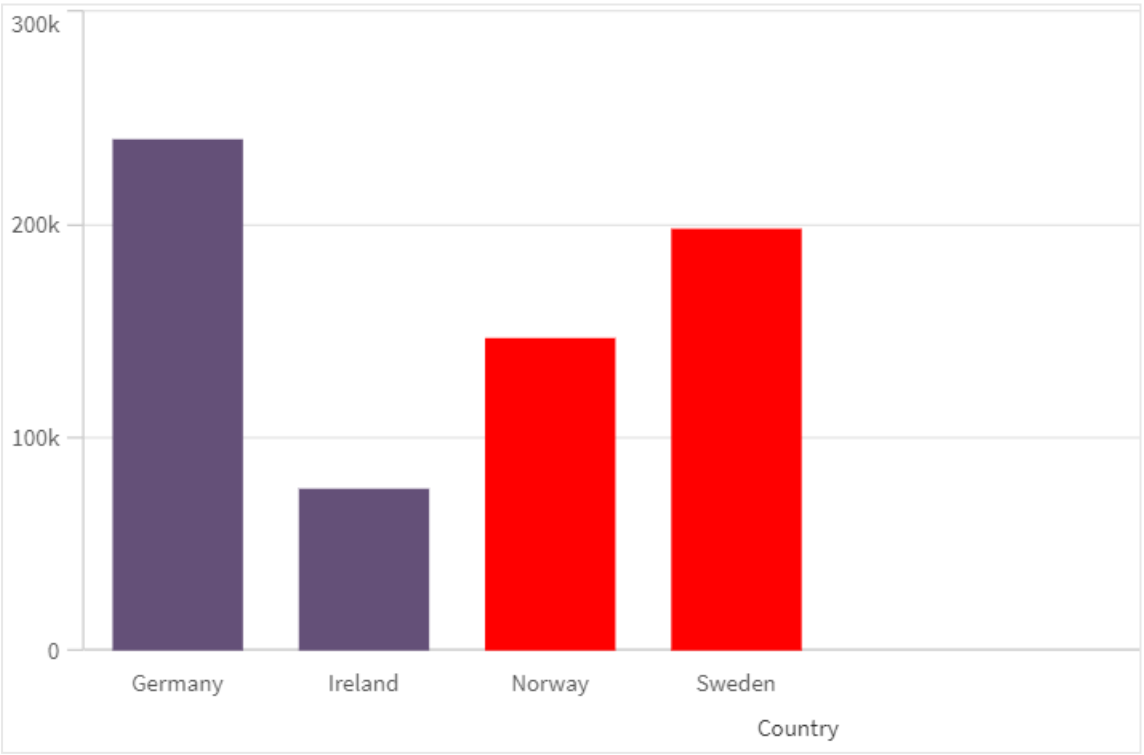
Dados usados neste exemplo:

```
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 ',');
```

Insira a seguinte expressão no painel de **propriedades** Cores e legenda:

```
If (Sum(Sales)>Sum(Budget),RGB(255,0,0),RGB(100,80,120))
```

Resultado:



Exemplo: Script de carregamento

O exemplo a seguir mostra os valores RGB equivalentes para valores em formato hexadecimal:

```
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];
```

Resultado:

Texto	Cor
010203	RGB(1,2,3)
AABBCC	RGB(170,187,204)

HSL

HSL() é utilizada em expressões para definir ou avaliar as propriedades de cores de um objeto gráfico, em que a cor é definida pelos valores de **hue**, **saturation** e **luminosity** entre 0 e 1.

Sintaxe:

```
HSL (hue, saturation, luminosity)
```

Tipo de dados de retorno: dual

Argumentos:

Argumentos

Argumento	Descrição
hue, saturation, luminosity	Valores de componenteshue, saturation e luminosity que variam entre 0 e 1.

Todos os argumentos devem ser expressões que resolvem para inteiros no intervalo de 0 a 1.

Se o componente numérico for interpretado e formatado em notação hexadecimal, a visualização dos valores de RGB dos componentes de cor torna-se mais fácil. Por exemplo, verde-claro tem o número 4 278 255 360 que, em notação hexadecimal, é FF00FF00 e RGB (0,255,0). Isso equivale a HSL (80/240, 240/240, 120/240) - um valor deHSL de (0.33, 1, 0.5).

5.5 Funções condicionais

As funções condicionais avaliam a condição e, em seguida, retornam respostas diferentes dependendo do valor da condição. As funções podem ser usadas no script de carregamento de dados e em expressões de gráfico.

Visão geral das funções condicionais

Cada função é descrita adicionalmente após a visão geral. Você também pode clicar no nome da função na sintaxe para acessar imediatamente os detalhes dessa função específica.

alt

A função **alt** retorna o primeiro parâmetro que tiver uma representação numérica válida. Se nenhuma correspondência for encontrada, o último parâmetro será retornado. É possível usar qualquer número de parâmetros.

```
alt (expr1[ , expr2 , expr3 , ...] , else)
```

class

A função **class** atribui o primeiro parâmetro a um intervalo de classe. O resultado é um valor duplo com a<=x<b como o valor textual, no qual a e b são os limites superior e inferior da caixa, e o limite inferior como valor numérico.

```
class (expression, interval [ , label [ , offset ]])
```

coalesce

A função **coalesce** retorna o primeiro dos parâmetros que possui uma representação válida de non-NULL. É possível usar qualquer número de parâmetros.

```
coalesce(expr1[ , expr2 , expr3 , ...])
```

if

A função **if** retorna um valor dependendo se a condição fornecida com a função avaliar como True ou False.

```
if (condition , then , else)
```

match

A função **match** compara o primeiro parâmetro com todos os seguintes e retorna a localização numérica das expressões correspondentes. A comparação faz distinção de maiúsculas e minúsculas.

```
match ( str, expr1 [ , expr2,...exprN ])
```

mixmatch

A função **mixmatch** compara o primeiro parâmetro com todos os seguintes e retorna a localização numérica das expressões correspondentes. A comparação não diferencia maiúsculas de minúsculas.

```
mixmatch ( str, expr1 [ , expr2,...exprN ])
```

pick
A função de escolha retorna a e*n*ésima expressão na lista.

```
pick (n, expr1[ , expr2,...exprN])
```

wildmatch
A função **wildmatch** compara o primeiro parâmetro com todos os seguintes e retorna o número da expressão correspondente. Permite o uso de caracteres curingas (***** e **?**) nas cadeias de comparação. ***** corresponde a qualquer sequência de caracteres. **?** corresponde a qualquer caractere simples. A comparação não diferencia maiúsculas de minúsculas.

```
wildmatch ( str, expr1 [ , expr2,...exprN ])
```

alt

A função **alt** retorna o primeiro parâmetro que tiver uma representação numérica válida. Se nenhuma correspondência for encontrada, o último parâmetro será retornado. É possível usar qualquer número de parâmetros.

Sintaxe:

```
alt(expr1[ , expr2 , expr3 , ...] , else)
```

Argumentos:

Argumentos

Argumento	Descrição
expr1	A primeira expressão na qual se buscará uma representação numérica válida.
expr2	A segunda expressão na qual se buscará uma representação numérica válida.
expr3	A terceira expressão na qual se buscará uma representação numérica válida.
else	Valor que será retornado, caso nenhum dos parâmetros anteriores tenha uma representação numérica válida.

A função alt é frequentemente usada com funções de interpretação de número ou data. Dessa forma, o Qlik Sense pode testar diferentes formatos de data em ordem de prioridade. Também pode ser usado para manipular os valores NULL em expressões numéricas.

Exemplos:

Exemplos

Exemplo	Resultado
`alt( date#( dat , 'YYYY/MM/DD' ), date#( dat , 'MM/DD/YYYY' ), date#( dat , 'MM/DD/YY' ), 'No valid date' )`	Essa expressão testa se o campo de data contém uma data de acordo com qualquer um dos três formatos de data especificados. Se sim, será retornado um valor duplo que contém a string original e uma representação numérica válida de data. Se nenhuma correspondência for encontrada, o texto 'No valid date' será retornado (sem nenhuma representação numérica válida).
`alt(Sales,0) + alt(Margin,0)`	Esta expressão inclui os campos Sales e Margin, substituindo qualquer valor ausente (NULL) por 0.

class

A função **class** atribui o primeiro parâmetro a um intervalo de classe. O resultado é um valor duplo com a<=x<b como o valor textual, no qual a e b são os limites superior e inferior da caixa, e o limite inferior como valor numérico.

Sintaxe:

```
class(expression, interval [ , label [ , offset ]])
```

Argumentos:

Argumentos

Argumento	Descrição
interval	Um número que especifica a largura da caixa.
label	Uma string arbitrária que pode substituir o “x” no texto do resultado.
offset	Um número que pode ser usado como deslocamento do ponto de partida padrão da classificação. O ponto de partida padrão é normalmente 0.

Exemplos:

Exemplos

Exemplo	Resultado
`class( var,10 )` com `var = 23`	retorna '20<=x<30'
`class( var,5,'value' )` com `var = 23`	retorna '20<= value <25'
`class( var,10,'x',5 )` com `var = 23`	retorna '15<=x<25'

Exemplo - Script de carregamento usando class

Exemplo: script de carregamento

Script de carregamento

Neste exemplo, carregamos uma tabela com o nome e idade de pessoas. O objetivo é incluir um campo que classifique cada pessoa de acordo com um grupo de idade, com dez anos de intervalo. A tabela de origem original tem a seguinte aparência.

Resultados

Name	Age
John	25
Karen	42
Yoshi	53

Para incluir o campo de classificação de faixa etária, você pode incluir a instrução de carregamento anterior através da função **class**.

Crie uma nova guia no editor de carregamento de dados e carregue os dados a seguir como um carregamento inline. Crie a tabela abaixo no Qlik Sense para ver os resultados.

```
LOAD *,
class(Age, 10, 'age') As Agegroup;

LOAD * INLINE
[ Age, Name
25, John
42, Karen
53, Yoshi];
```

Resultados

Resultados

Name	Age	Agegroup
John	25	20 <= age < 30
Karen	42	40 <= age < 50
Yoshi	53	50 <= age < 60

coalesce

A função **coalesce** retorna o primeiro dos parâmetros que possui uma representação válida de non-NULL. É possível usar qualquer número de parâmetros.

Sintaxe:

```
coalesce(expr1[ , expr2 , expr3 , ...])
```

Argumentos:

Argumentos

Argumento	Descrição
expr1	A primeira expressão para verificar se há uma representação não NULL válida.
expr2	A segunda expressão para verificar se há uma representação não NULL válida.
expr3	A terceira expressão para verificar se há uma representação não NULL válida.

Exemplos:

Exemplos

Exemplo	Resultado
	Essa expressão altera todos os valores NULL de um campo para "N/A".
`Coalesce(ProductDescription, ProductName, ProductCode, 'no description available')`	Essa expressão selecionará entre três campos de descrição do produto diferentes, para quando alguns campos podem não ter valores para o produto. O primeiro dos campos, na ordem especificada, com um valor não nulo será retornado. Se nenhum dos campos contiver um valor, o resultado será "nenhuma descrição disponível".
`Coalesce(TextBetween(FileName, '"', '"'), FileName)`	Essa expressão eliminará possíveis aspas de delimitação do campo *FileName*. Se o *FileName* fornecido estiver entre apas, essas serão removidas, e um *FileName* delimitado sem aspas será retornado. Se a função *TextBetween* não encontrar os delimitadores, ela retornará null, que **Coalesce** rejeitará, retornando em vez disso o *FileName* bruto.

if

A função **if** retorna um valor dependendo se a condição fornecida com a função avaliar como True ou False.

Sintaxe:

```
if(condition , then [, else])
```

Argumentos

Argumento	Descrição
condition	Expressão que é interpretada de forma lógica.

Argumento	Descrição
then	Expressão que pode ser de qualquer tipo. Se *condition* forTrue, então a função if retornará o valor da expressão *then*.
else	Expressão que pode ser de qualquer tipo. Se *condition* forFalse, então a função if retornará o valor da expressão *else*. Este parâmetro é opcional. Se *condition* for False, NULL será retornado se você não tiver especificado else.

Exemplo

Exemplo	Resultado
`if( Amount>= 0, 'OK', 'Alarm' )`	Essa expressão testa se o valor é um número positivo (0 ou maior) e retorna 'OK' se for. Se a quantidade for menor que 0, 'Alarm' é exibido.

Exemplo - Script de carregamento usando if

Exemplo: Script de carregamento

Script de carregamento

É possível usar If em um script de carregamento com outros métodos e objetos, incluindo variáveis. Por exemplo, se você definir uma variável *threshold* e quiser incluir um campo no modelo de dados com base nesse limite, poderá fazer o seguinte.

Crie uma nova guia no editor de carregamento de dados e carregue os dados a seguir como um carregamento inline. Crie a tabela abaixo no Qlik Sense para ver os resultados.

```
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;
```

Resultados

Tabela do Qlik Sense mostrando a saída do uso da função *if* no script de carregamento.

transaction_id	Comparado a Limite
3750	Menor que 100
3751	Maior que 100
3752	Menor que 100
3753	Maior que 100
3754	Maior que 100
3756	Menor que 100
3757	Maior que 100

Exemplos - Expressões de gráfico usando if

Exemplos: Expressões de gráfico

Expressão de gráfico 1

Script de carregamento

Crie uma nova guia no editor de carregamento de dados e carregue os dados a seguir como um carregamento inline. Depois de carregar os dados, crie os exemplos de expressão de gráfico abaixo em uma tabela do Qlik Sense.

```
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];
```

Tabela do Qlik Sense mostrando exemplos da função *if* função em uma expressão de gráfico.

Date	Localização	Incidentes	if(Incidentes>=10, 'Crítico', 'OK')	if(Incidentes>=10, 'Crítico', If(Incidentes>=1 and Incidentes<10, 'Aviso', 'OK'))
1/3/2016	Pequim	0	OK	OK
1/3/2016	Boston	12	Crítico	Crítico
1/3/2016	Estocolmo	3	OK	Aviso
1/3/2016	Toronto	0	OK	OK
1/4/2016	Pequim	0	OK	OK
1/4/2016	Boston	8	OK	Aviso

Expressão de gráfico 2

Em um novo aplicativo, adicione o script a seguir em uma nova guia no editor de carregamento de dados e carregue os dados. Você pode então criar a tabela com as expressões de gráfico abaixo.

```
SET FirstWeekDay=0;
Load
Date(MakeDate(2022)+RecNo()-1) as Date
Autogenerate 14;
```

Tabela do Qlik Sense mostrando um exemplo da função *if* em uma expressão de gráfico.

Date	WeekDay(Date)	If(WeekDay (Date)>=5,'WeekEnd','Normal Day')
1/1/2022	Sáb	Fim de semana
1/2/2022	Dom	Fim de semana
1/3/2022	Seg	Dia normal
1/4/2022	Ter	Dia normal
1/5/2022	Qua	Dia normal
1/6/2022	Qui	Dia normal
1/7/2022	Sex	Dia normal
1/8/2022	Sáb	Fim de semana
1/9/2022	Dom	Fim de semana
1/10/2022	Seg	Dia normal
1/11/2022	Ter	Dia normal
1/12/2022	Qua	Dia normal
1/13/2022	Qui	Dia normal

Date	WeekDay(Date)	If(WeekDay (Date)>=5,'WeekEnd','Normal Day')
1/14/2022	Sex	Dia normal

match

A função **match** compara o primeiro parâmetro com todos os seguintes e retorna a localização numérica das expressões correspondentes. A comparação faz distinção de maiúsculas e minúsculas.

Sintaxe:

```
match( str, expr1 [ , expr2,...exprN ])
```

*Se você quer usar comparação que não diferencia caracteres maiúsculos de minúsculos, use a função **mixmatch**. Se você quer usar comparação que não diferencia caracteres maiúsculos de minúsculos e "caracteres curingas", use a função **wildmatch**.*

Exemplo: Script de carregamento usando match

Exemplo: Script de carregamento

Script de carregamento

Você pode usar match para carregar um subconjunto de dados. Por exemplo, você pode retornar um valor numérico para uma expressão na função. Em seguida, é possível limitar os dados carregados com base no valor numérico. Match retornará 0 se não houver correspondência. Portanto, todas as expressões não correspondidas neste exemplo retornarão 0 e serão excluídas do carregamento de dados pelo comando WHERE.

Crie uma nova guia no editor de carregamento de dados e carregue os dados a seguir como um carregamento inline. Crie a tabela abaixo no Qlik Sense para ver os resultados.

```
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;
```

Resultados

Tabela do Qlik Sense mostrando a saída do uso da função match no script de carregamento

Color Code Blue and Black	Customer
Black	203521
Black	3036491
Blue	2038593

Exemplos - Expressões de gráfico usando match

Exemplos: Expressões de gráfico

Expressão de gráfico 1

Script de carregamento

Crie uma nova guia no editor de carregamento de dados e carregue os dados a seguir como um carregamento inline. Depois de carregar os dados, crie os exemplos de expressão de gráfico abaixo em uma tabela do Qlik Sense.

```
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];
```

A primeira expressão da tabela a seguir retorna 0 para Stockholm, porque "Stockholm" não está incluído na lista de expressões da função **match**. Também retorna 0 para "Zurich", porque a comparação **match** diferencia maiúsculas de minúsculas.

Tabela do Qlik Sense mostrando exemplos da função *match* em uma expressão de gráfico.

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

Expressão de gráfico 2

Você pode usar match para realizar uma classificação personalizada para uma expressão.

Por padrão, as colunas são classificadas numérica ou alfabeticamente, dependendo dos dados.

Tabela do Qlik Sense mostrando um exemplo da ordem de classificação padrão

Cities
Beijing
Boston
Stockholm
Toronto
zurich

Para alterar a ordem, faça o seguinte:

1. Abra a seção **Classificação** do gráfico no painel **Propriedades**.
2. Desative a classificação automática para a coluna na qual você deseja fazer uma classificação personalizada.
3. Desmarque **Classificar numericamente** e **Classificar alfabeticamente**.
4. Selecione **Classificar por expressão** e insira uma expressão semelhante à seguinte:

```
=match( Cities, 'Toronto','Boston','Beijing','Stockholm','zurich')
```
A ordem de classificação na coluna Cities é alterada.

Tabela do Qlik Sense mostrando um exemplo de alteração da ordem de classificação usando a função *match*

Cities
Toronto
Boston
Beijing
Stockholm
zurich

Você também pode exibir o valor numérico retornado.

Tabela do Qlik Sense mostrando um exemplo dos valores numéricos que são retornados da função *match*

Cidades	**Cities & ' - ' & match (Cities, 'Toronto','Boston', 'Beijing','Stockholm','zurich')**
Toronto	Toronto - 1
Boston	Boston - 2
Beijing	Beijing - 3
Stockholm	Stockholm - 4
zurich	zurich - 5

mixmatch

A função **mixmatch** compara o primeiro parâmetro com todos os seguintes e retorna a localização numérica das expressões correspondentes. A comparação não diferencia maiúsculas de minúsculas.

Sintaxe:

```
mixmatch( str, expr1 [ , expr2,...exprN ])
```
Se, em vez disso, você quiser usar a comparação sensível a maiúsculas e minúsculas, use a função **match**. Se você quer usar comparação que não diferencia caracteres maiúsculos de minúsculos e "caracteres curingas", use a função **wildmatch**.

Exemplo - Script de carregamento usando mixmatch

Exemplo: Script de carregamento

Script de carregamento

Você pode usar mixmatch para carregar um subconjunto de dados. Por exemplo, você pode retornar um valor numérico para uma expressão na função. Em seguida, é possível limitar os dados carregados com base no valor numérico. Mixmatch retornará 0 se não houver correspondência. Portanto, todas as expressões não correspondidas neste exemplo retornarão 0 e serão excluídas do carregamento de dados pelo comando WHERE.

Crie uma nova guia no editor de carregamento de dados e carregue os dados a seguir como um carregamento inline. Crie a tabela abaixo no Qlik Sense para ver os resultados.

```
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;
```

Resultados

Tabela do Qlik Sense mostrando a saída do uso da função mixmatch no script de carregamento.

Color Code Black, Blue, blue	Customer
Black	203521
Black	3036491
Blue	2038593
blue	5646471

Exemplos - Expressões de gráfico usando mixmatch

Exemplos: Expressões de gráfico

Crie uma nova guia no editor de carregamento de dados e carregue os dados a seguir como um carregamento inline. Depois de carregar os dados, crie os exemplos de expressão de gráfico abaixo em uma tabela do Qlik Sense.

Expressão de gráfico 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];
```

A primeira expressão da tabela a seguir retorna 0 para Stockholm, porque "Stockholm" não está incluído na lista de expressões da função **mixmatch**. Ela retorna 4 para "Zurich", porque a comparação **mixmatch** não diferencia maiúsculas de minúsculas.

Tabela do Qlik Sense mostrando exemplos da função *mixmatch* em uma expressão de gráfico

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

Expressão de gráfico 2

Você pode usar mixmatch para executar uma classificação personalizada para uma expressão.

Por padrão, as colunas são classificadas numérica ou alfabeticamente, dependendo dos dados.

Tabela do Qlik Sense mostrando um exemplo da ordem de classificação padrão

Cities
Beijing
Boston
Stockholm
Toronto
zurich

Para alterar a ordem, faça o seguinte:

1. Abra a seção **Classificação** do gráfico no painel **Propriedades**.
2. Desative a classificação automática para a coluna na qual você deseja fazer uma classificação personalizada.
3. Desmarque **Classificar numericamente** e **Classificar alfabeticamente**.
4. Selecione **Classificar por expressão** e insira a seguinte expressão:

```
=mixmatch( Cities, 'Toronto','Boston','Beijing','Stockholm','Zurich')
```

A ordem de classificação na coluna Cities é alterada.

Tabela Qlik Sense mostrando um exemplo de alteração da ordem de classificação usando a função *mixmatch*.

Cities
Toronto
Boston
Beijing
Stockholm
zurich

Você também pode exibir o valor numérico retornado.

Tabela Qlik Sense mostrando um exemplo dos valores numéricos que são retornados da função *mixmatch*.

Cidades	**Cities & ' - ' & mixmatch (Cities, 'Toronto','Boston', 'Beijing','Stockholm','Zurich')**
Toronto	Toronto - 1
Boston	Boston - 2
Beijing	Beijing - 3
Stockholm	Stockholm - 4
zurich	zurich - 5

pick

A função de escolha retorna a e*n*ésima expressão na lista.

Sintaxe:

```
pick(n, expr1[ , expr2,...exprN])
```

Argumentos:

Argumentos

Argumento	**Descrição**
n	*n*é um inteiro entre 1 e N.

Exemplo:

Exemplo

Exemplo	Resultado
`pick( N, 'A','B',4, 6 )`	retorna 'B' se N = 2 retorna 4 se N = 3

wildmatch

A função **wildmatch** compara o primeiro parâmetro com todos os seguintes e retorna o número da expressão correspondente. Permite o uso de caracteres curingas (***** e **?**) nas cadeias de comparação. ***** corresponde a qualquer sequência de caracteres. **?** corresponde a qualquer caractere simples. A comparação não diferencia maiúsculas de minúsculas.

Sintaxe:

```
wildmatch( str, expr1 [ , expr2,...exprN ])
```

Se você quer usar comparação sem caracteres curingas, use a função **match** ou **mixmatch**.

Exemplo: Script de carregamento usando wildmatch

Exemplo: Script de carregamento

Script de carregamento

Você pode usar wildmatch para carregar um subconjunto de dados. Por exemplo, você pode retornar um valor numérico para uma expressão na função. Em seguida, é possível limitar os dados carregados com base no valor numérico. Wildmatch retornará 0 se não houver correspondência. Portanto, todas as expressões não correspondidas neste exemplo retornarão 0 e serão excluídas do carregamento de dados pelo comando WHERE.

Crie uma nova guia no editor de carregamento de dados e carregue os dados a seguir como um carregamento inline. Crie a tabela abaixo no Qlik Sense para ver os resultados.

```
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;
```

Resultados

Tabela do Qlik Sense mostrando a saída do uso da função *wildmatch* no script de carregamento

Color Code Black, Blue, blue, Red	Customer
Black	203521
Black	3036491
Blue	2038593
blue	5646471
Red	049681
Red	2038593

Exemplos: Expressões de gráfico usando wildmatch

Exemplo: Expressão de gráfico

Expressão de gráfico 1

Crie uma nova guia no editor de carregamento de dados e carregue os dados a seguir como um carregamento inline. Depois de carregar os dados, crie os exemplos de expressão de gráfico abaixo em uma tabela do Qlik Sense.

```
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];
```

A primeira expressão da tabela a seguir retorna 0 para Stockholm, porque "Stockholm" não está incluído na lista de expressões da função **wildmatch**. Também retorna 0 para 'Boston', porque ? só corresponde em um caractere.

Tabela do Qlik Sense mostrando exemplos da função *wildmatch* em uma expressão de gráfico

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

Expressão de gráfico 2

Você pode usar wildmatch para executar uma classificação personalizada para uma expressão.

Por padrão, as colunas são classificadas numérica ou alfabeticamente, dependendo dos dados.

Tabela do Qlik Sense mostrando um exemplo da ordem de classificação padrão

Cities
Beijing
Boston
Stockholm
Toronto
zurich

Para alterar a ordem, faça o seguinte:

1. Abra a seção **Classificação** do gráfico no painel **Propriedades**.
2. Desative a classificação automática para a coluna na qual você deseja fazer uma classificação personalizada.
3. Desmarque **Classificar numericamente** e **Classificar alfabeticamente**.
4. Selecione **Classificar por expressão** e insira uma expressão semelhante à seguinte:
   `=wildmatch( Cities, 'Tor*','???ton','Beijing','Stockholm','*urich')`
   A ordem de classificação na coluna Cities é alterada.

Tabela Qlik Sense mostrando um exemplo de alteração da ordem de classificação usando a função *wildmatch*.

Cities
Toronto
Boston
Beijing
Stockholm
zurich

Você também pode exibir o valor numérico retornado.

Tabela do Qlik Sense mostrando um exemplo dos valores numéricos que são retornados da função *wildmatch*

Cidades	**Cities & ' - ' & wildmatch (Cities, 'Tor*','???ton','Beijing','Stockholm','*urich')**
Toronto	Toronto - 1
Boston	Boston - 2

Cidades	Cities & ' - ' & wildmatch (Cities, 'Tor*','???ton','Beijing','Stockholm','*urich')
Beijing	Beijing - 3
Stockholm	Stockholm - 4
zurich	zurich - 5

5.6 Funções de contador

Esta seção descreve as funções relacionadas aos contadores de registros durante a avaliação do comando **LOAD** no script de carregamento de dados. A única função que pode ser usada em expressões do gráfico é **RowNo()**.

Algumas funções do contador não têm parâmetros, mas os parênteses à direita ainda são necessários.

Visão geral das funções de contador

Cada função é descrita adicionalmente após a visão geral. Você também pode clicar no nome da função na sintaxe para acessar imediatamente os detalhes dessa função específica.

autonumber
Essa função de script retorna um valor inteiro exclusivo para cada valor distinto avaliado de *expression* encontrado durante a execução do script. Esta função pode ser usada, por exemplo, para criar uma representação de memória compacta de uma chave composta.

```
autonumber (expression[ , AutoID])
```

autonumberhash128
Essa função de script calcula um hash de 128 bits dos valores combinados de entrada de expressão e retorna um valor inteiro exclusivo para cada valor de hash distinto encontrado durante a execução do script. Esta função pode ser usada para criar uma representação de memória compacta de uma chave composta.

```
autonumberhash128 (expression {, expression})
```

autonumberhash256
Essa função de script calcula um hash de 256 bits dos valores combinados de entrada da expressão e retorna um valor inteiro exclusivo para cada valor de hash distinto encontrado durante a execução do script. Esta função pode ser usada, por exemplo, para criar uma representação de memória compacta de uma chave composta.

```
autonumberhash256 (expression {, expression})
```

IterNo
Essa função de script retorna um inteiro que indica o número de vezes que o registro foi carregado de acordo com a quantidade definida em uma declaração **LOAD** com uma cláusula **while**. A primeira iteração tem o número 1. A função **IterNo** somente será significativa se for usada junto com uma cláusula **while**.

```
IterNo ( )
```

RecNo
Essa função de script retorna um inteiro do número da linha lida atualmente da tabela atual. O primeiro registro é o número 1.

```
RecNo ( )
```

RowNo - script function
Esta função retorna um número inteiro da posição da linha atual na tabela interna resultante do Qlik Sense. A primeira linha é o número 1.

```
RowNo ( )
```

RowNo - chart function
RowNo() retorna o número da linha atual no atual segmento de coluna em uma tabela. Para gráficos de bitmap, **RowNo()** retorna o número da linha atual no equivalente de tabela estática do gráfico.

```
RowNo - função de gráfico([TOTAL])
```

autonumber

Essa função de script retorna um valor inteiro exclusivo para cada valor distinto avaliado de *expression* encontrado durante a execução do script. Esta função pode ser usada, por exemplo, para criar uma representação de memória compacta de uma chave composta.

> *Você só pode conectar chaves **autonumber** que foram geradas na mesma carga de dados, enquanto que um inteiro é gerado de acordo com a ordem em que a tabela é lida. Se você precisa usar chaves que são persistentes entre carga de dados, independente de triagem de fontes de dados, você deve usar as funções **hash128**, **hash160** ou **hash256**.*

Sintaxe:

```
autonumber(expression[ , AutoID])
```

Argumentos:

Argumento	Descrição
AutoID	Para criar várias instâncias de contador, se a função **autonumber** for usada em chaves diferentes no script, o parâmetro opcional *AutoID* poderá ser usado para nomear cada contador.

Exemplo: Criando uma chave composta

Neste exemplo, criamos uma chave composta usando a função **autonumber** para preservar a memória. O exemplo é breve para fins de demonstração, mas seria significativo com uma tabela contendo um grande número de linhas.

Dados de exemplo

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

Os dados de origem são carregados usando dados inline. Em seguida, adicione uma carga anterior, que cria uma chave composta a partir dos campos Region, Year e Month.

```
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
];
```

A tabela resultante tem a seguinte aparência:

Tabela de resultados

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

Neste exemplo, você pode consultar a RYMkey, para o exemplo 1, em vez de a string 'North2014May', se você precisa conectar-se a uma outra tabela.

Agora, carregaremos uma tabela de origem de custos de uma forma similar. Os campos Region, Year e Month são excluídos da carga anterior para evitar a criação de uma chave sintética, já estamos criando uma chave composta com a função **autonumber**, ligando as tabelas.

```
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
];
```

Agora, podemos adicionar uma visualização de tabela a uma pasta e adicionar os campos Region, Year e Month, bem como as medidas de Soma para vendas e custos. A tabela terá a seguinte aparência:

Tabela de resultados

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

autonumberhash128

Essa função de script calcula um hash de 128 bits dos valores combinados de entrada de expressão e retorna um valor inteiro exclusivo para cada valor de hash distinto encontrado durante a execução do script. Esta função pode ser usada para criar uma representação de memória compacta de uma chave composta.

*Você só pode conectar chaves **autonumberhash128** que foram geradas na mesma carga de dados, enquanto que um inteiro é gerado de acordo com a ordem em que a tabela é lida. Se você precisa usar chaves que são persistentes entre carga de dados, independente de triagem de fontes de dados, você deve usar as funções **hash128**, **hash160** ou **hash256**.*

Sintaxe:

```
autonumberhash128(expression {, expression})
```

Exemplo: Criando uma chave composta

Neste exemplo, criamos uma chave composta usando a função **autonumberhash128** para preservar a memória. O exemplo é breve para fins de demonstração, mas seria significativo com uma tabela contendo um grande número de linhas.

Dados de exemplo

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

Os dados de origem são carregados usando dados inline. Em seguida, adicione uma carga anterior, que cria uma chave composta a partir dos campos Region, Year e Month.

```
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
];
```

A tabela resultante tem a seguinte aparência:

Tabela de resultados

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

Neste exemplo, você pode consultar a RYMkey, para o exemplo 1, em vez de a string 'North2014May', se você precisa conectar-se a uma outra tabela.

Agora, carregaremos uma tabela de origem de custos de uma forma similar. Os campos Region, Year e Month são excluídos da carga anterior para evitar a criação de uma chave sintética, já estamos criando uma chave composta com a função **autonumberhash128**, ligando as tabelas.

```
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
];
```

Agora, podemos adicionar uma visualização de tabela a uma pasta e adicionar os campos Region, Year e Month, bem como as medidas de Soma para vendas e custos. A tabela terá a seguinte aparência:

Tabela de resultados

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

Essa função de script calcula um hash de 256 bits dos valores combinados de entrada da expressão e retorna um valor inteiro exclusivo para cada valor de hash distinto encontrado durante a execução do script. Esta função pode ser usada, por exemplo, para criar uma representação de memória compacta de uma chave composta.

*Você só pode conectar chaves **autonumberhash256** que foram geradas na mesma carga de dados, enquanto que um inteiro é gerado de acordo com a ordem em que a tabela é lida. Se você precisa usar chaves que são persistentes entre carga de dados, independente de triagem de fontes de dados, você deve usar as funções **hash128**, **hash160** ou **hash256**.*

Sintaxe:

```
autonumberhash256(expression {, expression})
```

Exemplo: Criando uma chave composta

Neste exemplo, criamos uma chave composta usando a função **autonumberhash256** para preservar a memória. O exemplo é breve para fins de demonstração, mas seria significativo com uma tabela contendo um grande número de linhas.

Tabela de exemplo

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

Os dados de origem são carregados usando dados inline. Em seguida, adicione uma carga anterior, que cria uma chave composta a partir dos campos Region, Year e Month.

```
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
];
```

A tabela resultante tem a seguinte aparência:

Tabela de resultados

Region	Year	Month	Sales	RYMkey
North	2014	May	245	1
North	2014	May	347	1
North	2014	June	127	2

Region	Year	Month	Sales	RYMkey
South	2014	June	645	3
South	2013	May	367	4
South	2013	May	221	4

Neste exemplo, você pode consultar a RYMkey, para o exemplo 1, em vez de a string 'North2014May', se você precisa conectar-se a uma outra tabela.

Agora, carregaremos uma tabela de origem de custos de uma forma similar. Os campos Region, Year e Month são excluídos da carga anterior para evitar a criação de uma chave sintética, já estamos criando uma chave composta com a função **autonumberhash256**, ligando as tabelas.

```
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
];
```

Agora, podemos adicionar uma visualização de tabela a uma pasta e adicionar os campos Region, Year e Month, bem como as medidas de Soma para vendas e custos. A tabela terá a seguinte aparência:

Tabela de resultados

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

Essa função de script retorna um inteiro que indica o número de vezes que o registro foi carregado de acordo com a quantidade definida em uma declaração **LOAD** com uma cláusula **while**. A primeira iteração tem o número 1. A função **IterNo** somente será significativa se for usada junto com uma cláusula **while**.

Sintaxe:

```
IterNo( )
```

Exemplos e resultados:

Exemplo:

```
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
];
```

Esse comando **LOAD** vai gerar um registro por data dentro do intervalo definido por **StartDate** e **EndDate**.

Nesse exemplo, a tabela resultante teria a seguinte aparência:

Tabela de resultados

Day	Date
1	2014-01-22
2	2014-01-23
3	2014-01-24
4	2014-01-25
5	2014-01-26

RecNo

Essa função de script retorna um inteiro do número da linha lida atualmente da tabela atual. O primeiro registro é o número 1.

Sintaxe:

```
RecNo( )
```

Ao contrário de **RowNo( )**, que conta as linhas na tabela resultante do Qlik Sense, **RecNo( )**, conta os registros na tabela de dados brutos e é restaurada quando uma tabela de dados brutos é concatenada a outra.

Exemplo: Script de carga de dados

Carregamento de tabelas de dados brutos:

```
Tab1:
LOAD * INLINE
```

```
[A, B
1, aa
2,cc
3,ee];

Tab2:
LOAD * INLINE
[C, D
5, xx
4,yy
6,zz];
```

Carregando registros e números de linha para as linhas selecionadas:

```
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;
```

A tabela interna resultante do Qlik Sense:

Tabela de resultados

A	B	RecNo( )	RowNo( )
1	aa	1	1
3	ee	3	2
4	yy	2	3
6	zz	3	4

RowNo

Esta função retorna um número inteiro da posição da linha atual na tabela interna resultante do Qlik Sense. A primeira linha é o número 1.

Sintaxe:

```
RowNo( [TOTAL])
```

Ao contrário de **RecNo( )**, que conta os registros na tabela de dados brutos, a função **RowNo( )** não conta os registros excluídos por cláusulas **where** e não é restaurada quando uma tabela de dados brutos é concatenada com outra.

*Se você usar o carregamento precedente, isto é, uma série de leituras de comandos **LOAD** empilhados da mesma tabela, poderá usar apenas **RowNo( )** no comando **LOAD** principal. Se você usar **RowNo( )** em comandos **LOAD** subsequentes, 0 é retornado.*

Exemplo: Script de carregamento de dados

Carregamento de tabelas de dados brutos:

```
Tab1:
LOAD * INLINE
[A, B
1, aa
2,cc
3,ee];

Tab2:
LOAD * INLINE
[C, D
5, xx
4,yy
6,zz];
```

Carregando registros e números de linha para as linhas selecionadas:

```
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;
```

A tabela interna resultante do Qlik Sense:

Tabela de resultados

A	B	RecNo( )	RowNo( )
1	aa	1	1
3	ee	3	2
4	yy	2	3
6	zz	3	4

RowNo - função de gráfico

RowNo() retorna o número da linha atual no atual segmento de coluna em uma tabela. Para gráficos de bitmap, **RowNo()** retorna o número da linha atual no equivalente de tabela estática do gráfico.

Se a tabela ou o equivalente de tabela tiver várias dimensões verticais, o segmento de coluna atual incluirá somente linhas com os mesmos valores que a linha atual em todas as colunas de dimensão, exceto na coluna que mostrar a última dimensão na ordem de classificação entre os campos.

Segmentos de coluna

A classificação por valores y em gráficos ou por colunas de expressão em tabelas não é permitida quando essa função de gráfico é usada em qualquer uma das expressões do gráfico. Essas alternativas de classificação estão, portanto, automaticamente desabilitadas. Quando você usar essa função de gráfico em uma visualização ou tabela, a classificação da visualização será revertida para a entrada classificada dessa função.

Sintaxe:

```
RowNo([TOTAL])
```

Tipo de dados de retorno: inteiro

Argumentos:

Argumento	Descrição
TOTAL	Se a tabela for unidimensional ou se o qualificador **TOTAL** for usado como argumento, o segmento de coluna atual será sempre igual à coluna inteira.

Exemplo: Expressão de gráfico usando RowNo

Exemplo - expressão de gráfico

Script de carregamento

Carregue os seguintes dados como um carregamento inline no editor de carregamento de dados para criar os exemplos de expressão de gráfico abaixo.

```
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 '|');
```

Expressão de gráfico

Crie uma visualização de tabela em uma pasta do Qlik Sense com **Customer** e **UnitSales** como dimensões. Adicione `RowNo( )` e `RowNo(TOTAL)` como medidas rotuladas **Linha no Segmento** e **Row Number**, respectivamente. Adicione a seguinte expressão à tabela como uma medida.

```
If( RowNo( )=1, 0, UnitSales / Above( UnitSales ))
```

Resultado

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

Customer	UnitSales	Row in Segment	Row Number	If(RowNo( )=1, 0, UnitSales / Above(UnitSales))
Divadip	1	1	9	0
Divadip	4	2	10	4

Explicação

A coluna **Row in Segment** mostra os resultados 1,2,3 para o segmento de coluna contendo os valores de UnitSales para cliente Astrida. Então, a numeração de coluna começa novamente em 1 para o próximo segmento de coluna, que é Betacab.

A coluna **Row Number** desconsidera as dimensões por causa do argumento TOTAL para RowNo() e conta as linhas da tabela.

Essa expressão retorna 0 para a primeira linha em cada segmento da coluna, fazendo com que a coluna mostre:

0, 2.25, 1.1111111, 0, 2.5, 5, 0, 2, 0 e 4.

Consulte também:

Above - função de gráfico (page 1282)

5.7 Funções de data e hora

As funções de data e hora do Qlik Sense são usadas para transformar e converter valores de data e hora. Todas as funções podem ser usadas no script de carregamento de dados e em expressões de gráfico.

As funções de data e hora têm como base um número de série de data e hora que é igual ao número de dias desde 30 de dezembro de 1899. O valor inteiro representa o dia e o valor fracionado representa a hora do dia.

O Qlik Sense usa o valor numérico do parâmetro, portanto, um número também é válido como parâmetro quando não está formatado como uma data ou hora. Se o parâmetro não corresponder ao valor numérico, por exemplo, se ele for uma string, então o Qlik Sense tenta interpretar a string de acordo com as variáveis de ambiente de data e hora.

Se o formato de hora usado no parâmetro não corresponder ao definido nas variáveis de ambiente, o Qlik Sense não poderá fazer uma interpretação correta. Para solucionar isso, altere a configuração ou use uma função de interpretação.

Nos exemplos para cada função, são considerados os formatos de data e hora padrão hh:mm:ss e YYYY-MM-DD (ISO 8601).

Ao processar um carimbo de data/hora com uma função de data ou hora, o Qlik Sense ignora os parâmetros de horário de verão, a menos que a função de data ou hora inclua uma posição geográfica.

Por exemplo, `ConvertToLocalTime( filetime('Time.qvd'), 'Paris')` usa parâmetros de horário de verão enquanto `ConvertToLocalTime(filetime('Time.qvd'), 'GMT-01:00')` não usa esses parâmetros.

Visão geral de funções de data e hora

Cada função é descrita adicionalmente após a visão geral. Você também pode clicar no nome da função na sintaxe para acessar imediatamente os detalhes dessa função específica.

Expressões com inteiro de hora

second

Esta função retorna um número inteiro que representa o segundo em que a fração da **expression** é interpretada como uma hora, de acordo com a interpretação numérica padrão.

```
second (expression)
```

minute

Esta função retorna um número inteiro que representa o minuto em que a fração da **expression** é interpretada como uma hora, de acordo com a interpretação numérica padrão.

```
minute (expression)
```

hour

Esta função retorna um número inteiro que representa a hora em que a fração da **expression** é interpretada como uma hora, de acordo com a interpretação numérica padrão.

```
hour (expression)
```

day

Esta função retorna um número inteiro que representa o dia em que a fração da **expression** é interpretada como uma data, de acordo com a interpretação numérica padrão.

```
day (expression)
```

week

Esta função retorna um número inteiro que representa o número da semana de acordo com a ISO 8601. O número da semana é calculado a partir da data de interpretação da expressão, de acordo com a interpretação de números padrão.

```
week (expression)
```

month
Esta função retorna um valor dual: um nome do mês, conforme definido na variável de ambiente **MonthNames** e um inteiro entre 1 e 12. O mês é calculado a partir da data de interpretação da expressão, de acordo com a interpretação do número padrão.

```
month (expression)
```

year
Esta função retorna um número inteiro que representa o ano em que a **expression** é interpretada como uma data, de acordo com a interpretação de número padrão.

```
year (expression)
```

weekyear
Essa função retorna o ano ao qual o número da semana pertence, de acordo com as variáveis de ambiente. O número da semana varia entre 1 e cerca de 52.

```
weekyear (expression)
```

weekday
Esta função retorna um valor dual com:

- Um nome do dia conforme definido na variável de ambiente **DayNames**.
- Um inteiro entre 0 e 6 correspondendo ao dia nominal da semana (0-6).

```
weekday (date)
```

Funções de data/hora

now
Essa função retorna um carimbo de data/hora da hora atual. A função retorna valores no formato da variável do sistema **TimeStamp**. O valor padrão de **timer_mode** é 1.

```
now ([ timer_mode])
```

today
Essa função retorna a data atual. A função retorna valores no formato da variável do sistema `DateFormat`.

```
today ([timer_mode])
```

LocalTime
Essa função retorna um carimbo de data/hora da hora atual para um fuso horário especificado.

```
localtime ([timezone [, ignoreDST ]])
```

Fazer funções

makedate
Esta função retorna uma data calculada a partir do ano **YYYY**, do mês **MM** e do dia **DD**.

```
makedate (YYYY [ , MM [ , DD ] ])
```

makeweekdate
Essa função retorna uma data calculada a partir do ano, do número da semana e do dia da semana.

```
makeweekdate (YYYY [ , WW [ , D ] ])
```

maketime
Esta função retorna um momento calculado a partir da hora **hh**, do minuto **mm** e do segundo **ss**.

```
maketime (hh [ , mm [ , ss [ .fff ] ] ])
```

Outras funções de data

AddMonths
Esta função retorna a data que ocorre **n** meses após a **startdate** ou, se **n** for negativo, a data que ocorre **n** meses antes da **startdate**.

```
addmonths (startdate, n , [ , mode])
```

AddYears
Essa função retorna a data ocorrendo **n** anos após **startdate** ou, se **n** for negativo, a data ocorrendo **n** anos antes de **startdate**.

```
addyears (startdate, n)
```

yeartodate
Esta função descobre se o carimbo de data e hora de entrada cai dentro de um ano da data em que o script foi carregado pela última vez, e retorna True se cair, False se não cair.

```
yeartodate (date [ , yearoffset [ , firstmonth [ , todaydate] ] ])
```

Funções de fuso horário

timezone
Esta função retorna o fuso horário, conforme definido no computador onde o mecanismo do Qlik está executando.

```
timezone ( )
```

GMT
Essa função retorna o Greenwich Mean Time atual, derivado das configurações regionais.

```
GMT ( )
```

UTC
Retorna o atual Coordinated Universal Time.

```
UTC ( )
```

daylightsaving
Retorna os ajustes atuais do horário de verão conforme definido no Windows.

```
daylightsaving ( )
```

converttolocaltime
Converte um carimbo de data/hora de UTC ou GMT para a hora local, na forma de um valor duplo. O lugar pode ser qualquer quantidade de cidades, lugares e fusos horários do mundo.

```
converttolocaltime (timestamp [, place [, ignore_dst=false]])
```

Definir funções de hora

setdateyear
Esta função admite como entrada um **timestamp** e **year** e atualiza o **timestamp** com o **year** especificado na entrada.

```
setdateyear  (timestamp, year)
```

setdateyearmonth
Esta função admite como entrada um **timestamp**, um **month** e um **year** , e atualiza o **timestamp** com o **year** e o **month** especificado na entrada.

```
setdateyearmonth  (timestamp, year, month)
```

Funções In...

inyear
Essa função retornará True se **timestamp** estiver dentro do ano que contém a **base_date**.

```
inyear (date, basedate , shift [, first_month_of_year = 1])
```

inyeartodate
Esta função retornará True se **timestamp** estiver na parte da parte do ano que contém **base_date** até e inclusive o último milissegundo de **base_date**.

```
inyeartodate  (date, basedate , shift [, first_month_of_year = 1])
```

inquarter
Esta função retornará True se **timestamp** estiver dentro do trimestre que contém **base_date**.

```
inquarter  (date, basedate , shift [, first_month_of_year = 1])
```

inquartertodate
Esta função retornará True se **timestamp** estiver na parte do trimestre que contém a **base_date** até e inclusive o último milissegundo de **base_date**.

```
inquartertodate  (date, basedate , shift [, first_month_of_year = 1])
```

inmonth
Esta função retornará True se **timestamp** estiver dentro do mês que contém **base_date**.

```
inmonth (date, basedate , shift)
```

inmonthtodate
Retornará True se **date** estiver na parte do mês que contém a **basedate** até e inclusive o último milissegundo da **basedate**.

```
inmonthtodate (date, basedate , shift)
```

inmonths
Essa função descobre se um carimbo de data/hora está dentro do mesmo mês, bimestre, trimestre, quadrimestre ou semestre como data base. Também é possível descobrir se uma data/hora está dentro de um período de tempo anterior ou seguinte.

```
inmonths (n, date, basedate , shift [, first_month_of_year = 1])
```

inmonthstodate
Essa função descobre se um carimbo de data/hora está dentro da parte de um período do mês, bimestre, trimestre, quadrimestre ou semestre, até e incluindo o último milissegundo de base_date. Também é possível descobrir se uma data/hora está dentro de um período de tempo anterior ou seguinte.

```
inmonthstodate (n, date, basedate , shift [, first_month_of_year = 1])
```

inweek
Essa função retornará True se **timestamp** estiver dentro da semana que contém **base_date**.

```
inweek (date, basedate , shift [, weekstart])
```

inweektodate
Essa função retornará True se **timestamp** estiver na parte da semana que contém a **base_date** até e inclusive o último milissegundo da **base_date**.

```
inweektodate (date, basedate , shift [, weekstart])
```

inlunarweek
Essa função determina se **timestamp** está dentro da semana lunar que contém **base_date**. As semanas lunares no Qlik Sense são definidas contando 1º de janeiro como o primeiro dia da semana. Além da última semana do ano, cada semana conterá exatamente sete dias.

```
inlunarweek (date, basedate , shift [, weekstart])
```

inlunarweektodate
Esta função descobre se **timestamp** está dentro da parte da semana lunar até e inclusive o último milissegundo da **base_date**. As semanas lunares no Qlik Sense são definidas contando 1º de janeiro como o primeiro dia da semana e, além da última semana do ano, conterão exatamente sete dias.

```
inlunarweektodate (date, basedate , shift [, weekstart])
```

inday
Esta função retorna True se **timestamp** estiver dentro do dia que contém **base_timestamp**.

```
inday (timestamp, basetimestamp , shift [, daystart])
```

indaytotime
Esta função retorna True se **timestamp** estiver na parte do dia que contém a **base_timestamp** até e inclusive o exato milissegundo da **base_timestamp**.

```
indaytotime (timestamp, basetimestamp , shift [, daystart])
```

Funções Start ... end

yearstart
Esta função retorna um carimbo de data/hora correspondente ao início do primeiro dia do ano que contém a **date**. O formato de saída padrão será o **DateFormat** definido no script.

```
yearstart ( date [, shift = 0 [, first_month_of_year = 1]])
```

yearend
Esta função retorna um valor correspondente a um carimbo de data/hora com o último milissegundo do último dia do ano que contém **date**. O formato de saída padrão será o **DateFormat** definido no script.

```
yearend ( date [, shift = 0 [, first_month_of_year = 1]])
```

yearname
Esta função retorna um ano com quatro dígitos como valor de exibição com um valor numérico subjacente que corresponde a um carimbo de data/hora com o primeiro milissegundo do primeiro dia do ano que contém **date**.

```
yearname (date [, shift = 0 [, first_month_of_year = 1]] )
```

quarterstart
Esta função retorna um valor correspondente a um carimbo de data/hora com o primeiro milissegundo do trimestre que contém **date**. O formato de saída padrão será o **DateFormat** definido no script.

```
quarterstart (date [, shift = 0 [, first_month_of_year = 1]])
```

quarterend
Esta função retorna um valor correspondente a uma data/hora com o último milissegundo do trimestre que contém **date**. O formato de saída padrão será o **DateFormat** definido no script.

```
quarterend (date [, shift = 0 [, first_month_of_year = 1]])
```

quartername
Esta função retorna um valor de exibição que mostra os meses do trimestre (formatados de acordo com a variável de script **MonthNames**) e o ano com um valor numérico subjacente que corresponde a um carimbo de hora do primeiro milissegundo do primeiro dia do trimestre.

```
quartername (date [, shift = 0 [, first_month_of_year = 1]])
```

monthstart
Esta função retorna um valor correspondente à data/hora com o primeiro milissegundo do primeiro dia do mês que contém **date**. O formato de saída padrão será o **DateFormat** definido no script.

```
monthstart (date [, shift = 0])
```

monthend
Esta função retorna um valor correspondente ao carimbo de data/hora do último milissegundo do último dia do mês que contém `date`. O formato de saída padrão será o `DateFormat` definido no script.

```
monthend (date [, shift = 0])
```

monthname
Esta função retorna um valor de exibição que mostra o mês (formatado de acordo com a variável de script **MonthNames**) e o ano com um valor numérico subjacente que corresponde a um carimbo de hora do primeiro milissegundo do primeiro dia do mês.

```
monthname (date [, shift = 0])
```

monthsstart
Essa função retorna um valor correspondente ao carimbo de data/hora do primeiro milissegundo do mês, bimestre, trimestre, quadrimestre ou semestre contendo uma data base. Também é possível descobrir a data/hora de um período de tempo anterior ou seguinte.O formato de saída padrão é o **DateFormat** definido no script.

```
monthsstart (n, date [, shift = 0 [, first_month_of_year = 1]])
```

monthsend
Essa função retorna um valor correspondente a um carimbo de data/hora do último milissegundo do mês, bimestre, trimestre, quadrimestre ou semestre contendo uma data base. Também é possível descobrir a data/hora de um período de tempo anterior ou seguinte.

```
monthsend (n, date [, shift = 0 [, first_month_of_year = 1]])
```

monthsname
Esta função retorna um valor de exibição que representa o intervalo dos meses do período (formatados de acordo com a variável de script **MonthNames**) e o ano. O valor numérico subjacente corresponde a um carimbo de data/hora do primeiro milissegundo do mês, bimestre, trimestre, quadrimestre ou semestre contendo uma data base.

```
monthsname (n, date [, shift = 0 [, first_month_of_year = 1]])
```

weekstart
Esta função retorna um valor correspondente ao carimbo de data/hora com o primeiro milissegundo do primeiro dia da semana do calendário que contém a **date**. O formato de saída padrão é o **DateFormat** definido no script.

```
weekstart (date [, shift = 0 [,weekoffset = 0]])
```

weekend
Essa função retorna um valor correspondente a um carimbo de data/hora do último milissegundo do último dia da semana do calendário contendo **date**. O formato de saída padrão será o **DateFormat** definido no script.

```
weekend (date [, shift = 0 [,weekoffset = 0]])
```

weekname
Esta função retorna um valor que mostra o número do ano e da semana com um valor numérico subjacente que corresponde a um carimbo de hora do primeiro milissegundo do primeiro dia da semana que contém a **date**.

```
weekname (date [, shift = 0 [,weekoffset = 0]])
```

lunarweekstart
Essa função retorna um valor correspondente a um carimbo de data/hora do primeiro milissegundo do primeiro dia da semana lunar que contém **date**. As semanas lunares no Qlik Sense são definidas contando 1º de janeiro como o primeiro dia da semana e, além da última semana do ano, conterão exatamente sete dias.

```
lunarweekstart (date [, shift = 0 [,weekoffset = 0]])
```

lunarweekend
Essa função retorna um valor correspondente a um carimbo de data/hora do último milissegundo do último dia da semana lunar que contém **date**. As semanas lunares no Qlik Sense são definidas contando 1º de janeiro como o primeiro dia da semana e, além da última semana do ano, conterão exatamente sete dias.

```
lunarweekend (date [, shift = 0 [,weekoffset = 0]])
```

lunarweekname
Esta função retorna um valor de exibição que mostra o número do ano e da semana lunar que corresponde a data/hora do primeiro milissegundo do primeiro dia da semana lunar que contém a **date**. As semanas lunares no Qlik Sense são definidas contando 1º de janeiro como o primeiro dia da semana e, além da última semana do ano, conterão exatamente sete dias.

```
lunarweekname (date [, shift = 0 [,weekoffset = 0]])
```

daystart
Esta função retorna um valor correspondente a uma data/hora com o primeiro milissegundo do dia contido no argumento **time**. O formato de saída padrão será o **TimestampFormat** definido no script.

```
daystart (timestamp [, shift = 0 [, dayoffset = 0]])
```

dayend
Esta função retorna um valor correspondente a uma data/hora com o milissegundo final do dia contido em **time**. O formato de saída padrão será o **TimestampFormat** definido no script.

```
dayend (timestamp [, shift = 0 [, dayoffset = 0]])
```

dayname
Esta função retorna um valor que mostra a data com um valor numérico subjacente que corresponde a um carimbo de hora do primeiro milissegundo do dia que contém o **time**.

```
dayname (timestamp [, shift = 0 [, dayoffset = 0]])
```

Funções de numeração de dia

age
A função **age** retorna a idade no momento do **timestamp** (em anos completados) de alguém nascido em **date_of_birth**.

```
age (timestamp, date_of_birth)
```

networkdays
A função **networkdays** retorna o número de dias úteis (segunda-sexta) entre e inclusive a **start_date** e **end_date**, levando em conta qualquer **holiday** opcionalmente listado.

```
networkdays  (start:date, end_date {, holiday})
```

firstworkdate
A função **firstworkdate** retorna a última data inicial para obter o **no_of_workdays** (segunda-sexta) com término não posterior à **end_date** levando em conta os feriados opcionalmente listados. **end_date** e **holiday** devem ser datas ou carimbos de data/hora válidos.

```
firstworkdate (end_date, no_of_workdays {, holiday} )
```

lastworkdate
A função **lastworkdate** retorna a primeira data de término para obter **no_of_workdays** (de segunda a sexta-feira), se começar em **start_date**, levando em consideração qualquer **holiday** opcionalmente listado. **start_date** e **holiday** devem ser datas ou carimbos de data/hora válidos.

```
lastworkdate (start_date, no_of_workdays {, holiday})
```

daynumberofyear
Essa função calcula o número do dia do ano que estiver em um carimbo de data/hora. O cálculo é feito a partir do primeiro milissegundo do primeiro dia do ano, mas o primeiro mês pode ser deslocado.

```
daynumberofyear (date[,firstmonth])
```

daynumberofquarter
Esta função calcula o número do dia do trimestre em que um timestamp cai. Essa função é usada ao criar um calendário mestre.

```
daynumberofquarter (date[,firstmonth])
```

addmonths

Esta função retorna a data que ocorre **n** meses após a **startdate** ou, se **n** for negativo, a data que ocorre **n** meses antes da **startdate**.

Sintaxe:

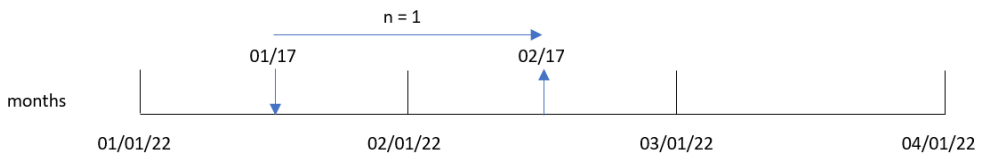
```
AddMonths(startdate, n , [ , mode])
```

Tipo de dados de retorno: dual

A função `addmonths()` adiciona ou subtrai um número definido de meses `n`, de uma `startdate` e retorna a data resultante.

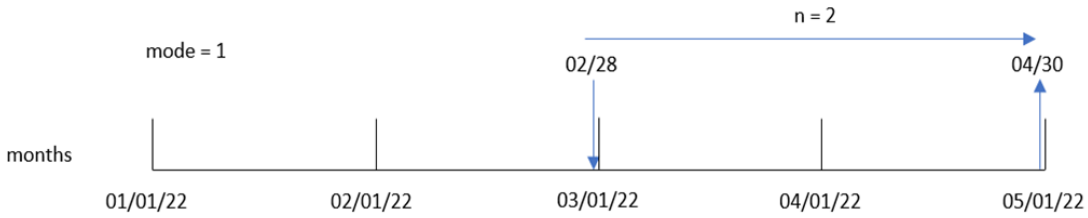
O argumento `mode` afetará os valores de `startdate` no dia 28 do mês ou depois dele. Ao definir o argumento `mode` como 1, a função `addmonths()` retorna uma data que é igual em distância relativa ao final do mês que `startdate`.

Exemplo de diagrama da função addmonths()



Por exemplo, 28 de fevereiro é o último dia do mês. Se a função addmonths(), com uma mode de 1, for usada para retornar a data dois meses depois, a função retornará a última data de abril, 30 de abril.

Exemplo de diagrama da função addmonths(), com mode=1



Argumentos

Argumento	Descrição
startdate	A data de início como um carimbo de data/hora, como “12/10/2012”.
n	Número de meses como um número inteiro positivo ou negativo.
mode	Especifica se o mês é adicionado em relação ao início ou ao final do mês. O modo padrão é 0 para adições relativas ao início do mês. Defina o modo como 1 para adições relativas ao final do mês. Quando o modo está definido como 1 e a data de entrada é 28 ou acima, a função verifica quantos dias faltam para chegar ao final do mês na data de início. O mesmo número de dias para chegar ao final do mês é definido na data retornada.

Quando usar

A função addmonths() normalmente será usada em uma expressão para encontrar uma data em um determinado número de meses antes ou depois de um período.

Por exemplo, a função addmonths() pode ser usada para identificar a data de término dos contratos de telefonia móvel.

Exemplos de funções

Exemplo	Resultado
addmonths ('01/29/2003' ,3)	Retorna "04/29/2003".
addmonths ('01/29/2003',3,0)	Retorna "04/29/2003".
addmonths ('01/29/2003',3,1)	Retorna "04/28/2003".

Exemplo	Resultado
addmonths ('01/29/2003',1,0)	Retorna "02/28/2003".
addmonths ('01/29/2003',1,1)	Retorna "02/26/2003".
addmonths ('02/28/2003',1,0)	Retorna "03/28/2003".
addmonths ('02/28/2003',1,1)	Retorna "03/31/2003".
addmonths ('01/29/2003',-3)	Retorna "10/29/2002".

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplo 1: Sem argumentos adicionais

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações entre 2020 e 2022, que é carregado em uma tabela denominada `Transactions`.
- O campo de data fornecido no formato da `variável` de sistema DateFormat (MM/DD/AAAA).
- A criação de um campo, `two_months_later`, que retorna a data de dois meses após a transação ter ocorrido.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `two_months_later`

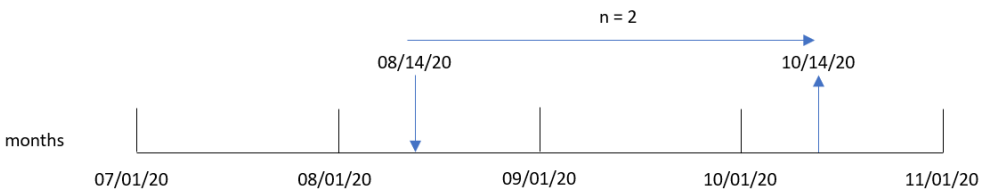
Tabela de resultados

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

date	two_months_later
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

O campo `two_months_later` é criado na instrução de carregamento anterior usando a função `addmonths()`. O primeiro argumento fornecido identifica qual data está sendo avaliada. O segundo argumento é o número de meses para adicionar a ou subtrair de `startdate`. Nesse caso, o valor de 2 é fornecido.

Diagrama da função `addmonths()`, exemplo sem argumentos adicionais



A transação 8193 ocorreu em 14 de agosto. Portanto, a função `addmonths()` retorna 14 de outubro de 2020 para o campo `two_months_later`.

Exemplo 2: Final relativo do mês

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações de fim de mês em 2022, que é carregado em uma tabela denominada `Transactions`.

- O campo de data fornecido no formato da variável de sistema `DateFormat` (MM/DD/AAAA).
- A criação de um campo, `relative_two_months_prior`, que retorna a data relativa de término do mês de dois meses antes da transação.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

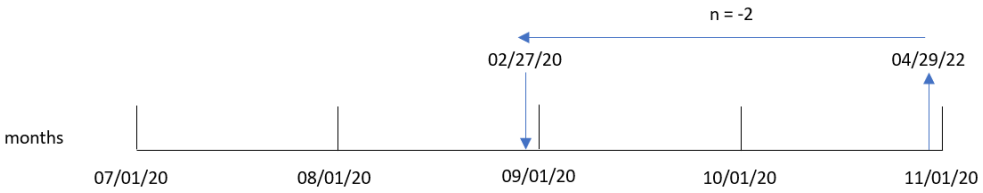
- `date`
- `relative_two_months_prior`

Tabela de resultados

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

O campo `relative_two_months_prior` é criado na instrução de carregamento anterior usando a função `addmonths()`. O primeiro argumento fornecido identifica qual data está sendo avaliada. O segundo argumento é o número de meses para adicionar a ou subtrair de `startdate`. Nesse caso, o valor de -2 é fornecido. O argumento final é o modo, com um valor de 1, que força a função a calcular a data relativa de fim de mês para todas as datas maiores que ou iguais a 28.

Diagrama da função `addmonths()`, exemplo com n=-2



A transação 8191 ocorre em 29 de abril de 2022. Inicialmente, dois meses antes definiria o mês como fevereiro. Então, como o terceiro argumento da função define o modo como 1 e o valor do dia é posterior ao dia 27, a função calcula o valor relativo do final do mês. A função identifica que 29 é o penúltimo dia de abril e, portanto, retorna o penúltimo dia de fevereiro, 27.

Exemplo 3: Exemplo de objeto de gráfico

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém o mesmo conjunto de dados e cenário do primeiro exemplo.

No entanto, neste exemplo, o conjunto de dados inalterado é carregado no aplicativo. O cálculo que retorna a data de dois meses após a transação ter ocorrido é criado como uma medida em um objeto de gráfico.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão: `date`.

Crie a seguinte medida:

```
=addmonths(date,2)
```

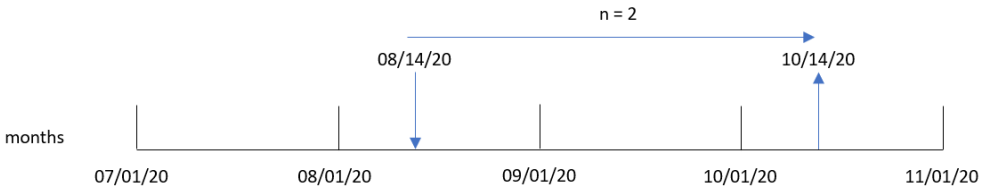
Tabela de resultados

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

date	=addmonths(date,2)
02/26/2022	04/26/2022
03/07/2022	05/07/2022
03/11/2022	05/11/2022

A medida `two_months_later` é criada no objeto de gráfico usando a função `addmonths()`. O primeiro argumento fornecido identifica qual data está sendo avaliada. O segundo argumento é o número de meses para adicionar a ou subtrair de `startdate`. Nesse caso, o valor de 2 é fornecido.

Diagrama da função `addmonths()`, exemplo de objeto de gráfico



A transação 8193 ocorreu em 14 de agosto. Portanto, a função `addmonths()` retorna 14 de outubro de 2020 para o campo `two_months_later`.

Exemplo 4: Cenário

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados que é carregado em uma tabela denominada `Mobile_Plans`.
- Informações com o ID do contrato, a data de início, a duração do contrato e a mensalidade.

O usuário final deseja um objeto de gráfico que mostre, por ID de contrato, a data de rescisão de cada contrato telefônico.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `contract_id`
- `start_date`
- `contract_length`

Crie a medida a seguir para calcular a data de término de cada contrato:

```
=addmonths(start_date,contract_length, 0)
```

Tabela de resultados

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

contract_id	start_date	contract_length	=addmonths(start_date,contract_length,0)
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

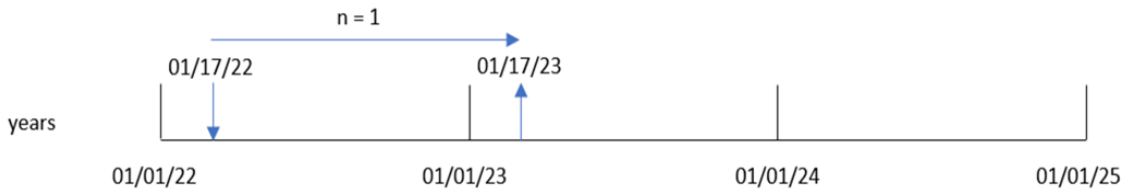
Essa função retorna a data ocorrendo **n** anos após **startdate** ou, se **n** for negativo, a data ocorrendo **n** anos antes de **startdate**.

Sintaxe:

```
AddYears(startdate, n)
```

Tipo de dados de retorno: dual

Exemplo de diagrama da função addyears()



A função addyears() soma ou subtrai um número definido de anos, n, de uma startdate. Em seguida, ela retorna a data resultante.

Argumentos

Argumento	Descrição
startdate	A data de início como um carimbo de data/hora, como “12/10/2012”.
n	Número de anos como um número inteiro positivo ou negativo.

Exemplos de funções

Exemplo	Resultado
`addyears ('01/29/2010',3)`	Retorna "01/29/2013".
`addyears ('01/29/2010',-1)`	Retorna "01/29/2009".

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplo 1: Exemplo simples

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações entre 2020 e 2022, que é carregado em uma tabela denominada `Transactions`.
- O campo de data fornecido no formato da `variável` de sistema DateFormat (MM/DD/AAAA).
- A criação de um campo, `two_years_later`, que retorna a data de dois anos após a transação ter ocorrido.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `two_years_later`

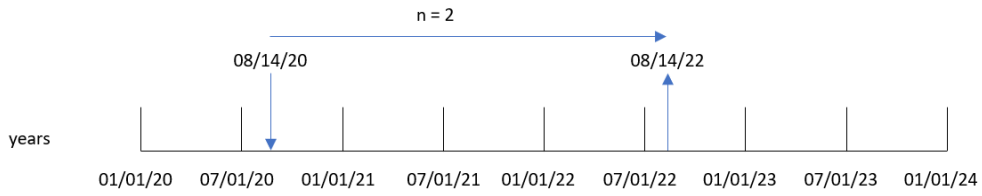
Tabela de resultados

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

date	two_years_later
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

O campo `two_years_later` é criado na instrução de carregamento anterior usando a função `addyears()`. O primeiro argumento fornecido identifica qual data está sendo avaliada. O segundo argumento é o número de anos a serem adicionados ou subtraídos da data de início. Nesse caso, o valor de 2 é fornecido.

Diagrama da função `addyears()`, exemplo básico



A transação 8193 ocorreu em 14 de agosto de 2020. Portanto, a função `addyears()` retorna 14 de agosto de 2022 para o campo `two_years_later`.

Exemplo 2: Exemplo de objeto de gráfico

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações entre 2020 e 2022, que é carregado em uma tabela chamada `Transactions`.
- O campo de data fornecido no formato da variável de sistema `DateFormat` (MM/DD/AAAA).

Em um objeto de gráfico, crie uma medida, `prior_year_date`, que retorna a data um ano antes da transação.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão: `date`.

Crie a medida a seguir para calcular a data um ano antes de cada transação:

```
=addyears(date,-1)
```

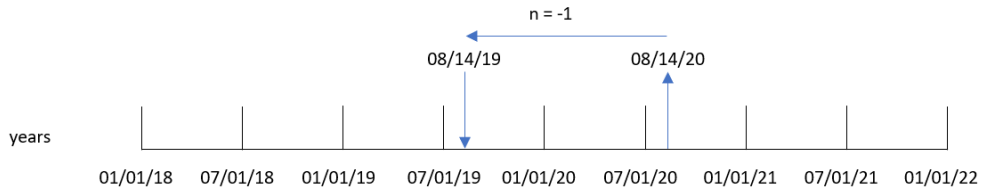
Tabela de resultados

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

date	=addyears(date,-1)
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
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

A medida `one_year_prior` é criada no objeto de gráfico usando a função `addyears()`. O primeiro argumento fornecido identifica qual data está sendo avaliada. O segundo argumento é o número de anos para adicionar ou subtrair de `startdate`. Nesse caso, o valor de -1 é fornecido.

Diagrama da função `addyears()`, exemplo de objeto de gráfico



A transação 8193 ocorreu em 14 de agosto. Portanto, a função `addyears()` retorna 14 de agosto de 2019 para o campo `one_year_prior`.

Exemplo 3: Cenário

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados que é carregado em uma tabela denominada `Warranties`.
- Informações com a ID do produto, a data de compra, a duração da garantia e o preço de compra.

O usuário final deseja um objeto de gráfico que mostre, por ID de produto, a data de rescisão da garantia de cada produto.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `product_id`
- `purchase_date`
- `warranty_length`

Crie a seguinte medida para calcular a data de término da garantia de cada produto:

```
=addyears(purchase_date,warranty_length)
```

Tabela de resultados

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
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

A função **age** retorna a idade no momento do **timestamp** (em anos completados) de alguém nascido em **date_of_birth**.

Sintaxe:

```
age(timestamp, date_of_birth)
```

Pde ser uma expressão.

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
timestamp	Data/hora, ou expressão que resulte em uma data/hora, para calcular o número total de anos.
date_of_birth	A data de nascimento de uma pessoa está sendo calculada. Pde ser uma expressão.

Exemplos e resultados:
Estes exemplos usam o formato de data **DD/MM/YYYY**. O formato de data é especificado no comando **SET DateFormat** na parte superior do seu script de carga de dados. Altere o formato nos exemplos para atender às suas necessidades.

Exemplos de scripts

Exemplo	Resultado
`age('25/01/2014', '29/10/2012')`	Returna 1.
`age('29/10/2014', '29/10/2012')`	Returna 2.

Exemplo:

Adicione o script de exemplo ao seu aplicativo e execute-o. Para ver o resultado, adicione os campos listados na coluna de resultados a uma pasta no seu aplicativo.

```
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;
```

A tabela resultante mostra os valores retornados de age para cada um dos registros na tabela.

Tabela de resultados

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
Sunny	14/5/1996	19
Ajoa	13/6/1996	19
Daphne	7/7/1998	17
Biffy	4/8/2000	15

converttolocaltime

Converte um carimbo de data/hora de UTC ou GMT para a hora local, na forma de um valor duplo. O lugar pode ser qualquer quantidade de cidades, lugares e fusos horários do mundo.

Sintaxe:

```
ConvertToLocalTime(timestamp [, place [, ignore_dst=false]])
```

Tipo de dados de retorno: dual

Argumentos

Argumento	Descrição
timestamp	O carimbo de data/hora ou expressão resolvendo para um carimbo de data/hora ser convertido.

Argumento	Descrição
place	Um local ou fuso horário da tabela de locais válidos e fusos horários abaixo. Opcionalmente, você pode usar as opções GMT ou UTC para definir a hora local. Os seguintes valores e intervalos de fuso horário são válidos: • 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 *Se você usar um deslocamento de horário de verão (ou seja, especificar um valor de argumento **ignore_dst** avaliando como False), deverá especificar um local, em vez de um deslocamento GMT, no argumento **place**. Isso ocorre porque o ajuste para o horário de verão requer informações latitudinais além das informações longitudinais fornecidas por um deslocamento GMT. Para obter informações, consulte Usando deslocamentos GMT em combinação com DST (page 628).* *Você pode usar somente fusos horários padrão. Não é possível usar um fuso horário arbitrário como, por exemplo, GMT-04:27.*
ignore_dst	Se este argumento for avaliado como True, DST (horário de verão) será ignorado. Valores de argumentos válidos avaliados para True incluem `-1` e `True()`. Se esse argumento for avaliado como False, o carimbo de data/hora será ajustado para o horário de verão. Valores de argumentos válidos avaliados para False incluem `0` e `False()`. Se o valor do argumento **ignore_dst** for inválido, a função avaliará a expressão como se o valor **ignore_dst** fosse avaliado como True. Se o valor do argumento **ignore_dst** não for especificado, a função avaliará a expressão como se o valor **ignore_dst** fosse avaliado como False.

Locais e fusos horários válidos

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

A-C	D-K	L-R	S-Z
Amsterdam	Edinburgh	Ljubljana	Sarajevo
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

A-C	D-K	L-R	S-Z
Central America	Kuala Lumpur	Pretoria	Zagreb
Central Time (US & Canada)	Kuwait	Quito	-
Chennai	Kyiv	Riga	-
Chihuahua	-	Riyadh	-
Chongqing	-	Rome	-
Copenhagen	-	-	-

Exemplos e resultados:

Exemplos de scripts

Exemplo	Resultado
`ConvertToLocalTime('2023-08-14 08:39:47','Paris')`	Retorna '2023-08-14 10:39:47' e a representação do carimbo de data/hora interno correspondente.
`ConvertToLocalTime(UTC(), 'Stockholm')`	Retorna a hora de Estocolmo, ajustando para o horário de verão.
`ConvertToLocalTime(UTC(), 'Stockholm', -1)`	Retorna a hora de Estocolmo, sem ajuste do horário de verão.
`ConvertToLocalTime(UTC(), 'GMT-05:00')`	Retorna a hora da costa leste norte-americana, por exemplo, Nova York. Nenhum ajuste é feito para o horário de verão porque um deslocamento GMT, em vez de um local, é especificado.
`ConvertToLocalTime(UTC(), 'New York', -1)`	Retorna a hora da costa leste da América do Norte (Nova York), sem ajuste do horário de verão.
`ConvertToLocalTime(UTC(), 'New York', True())`	Retorna a hora da costa leste da América do Norte (Nova York), sem ajuste do horário de verão.
`ConvertToLocalTime(UTC(), 'New York', 0)`	Retorna a hora da costa leste da América do Norte (Nova York), ajustando para o horário de verão.
`ConvertToLocalTime(UTC(), 'New York', False())`	Retorna a hora da costa leste da América do Norte (Nova York), ajustando para o horário de verão.

Usando deslocamentos GMT em combinação com DST

Após a implementação das bibliotecas de Componentes Internacionais para Unicode (ICU) no Qlik Sense, o uso de deslocamentos GMT (Greenwich Mean Time) em combinação com DST (Daylight Saving Time) requer informações latitudinais adicionais.

GMT é um deslocamento longitudinal (leste-oeste), enquanto DST é um deslocamento latitudinal (norte-sul). Por exemplo, Helsinque (Finlândia) e Joanesburgo (África do Sul) compartilham o mesmo deslocamento GMT+02:00, mas não compartilham o mesmo deslocamento DST. Isso significa que, além do deslocamento GMT, qualquer deslocamento DST requer informações sobre a posição latitudinal do fuso horário local (entrada de fuso horário geográfico) para obter informações completas sobre as condições locais do DST.

day

Esta função retorna um número inteiro que representa o dia em que a fração da **expression** é interpretada como uma data, de acordo com a interpretação numérica padrão.

A função retorna o dia do mês para uma data específica. Ela é geralmente usada para derivar um campo de dia como parte de uma dimensão de calendário.

Sintaxe:

```
day(expression)
```

Tipo de dados de retorno: inteiro

Exemplos de funções

Exemplo	Resultado
`day( 1971-10-12 )`	retorna 12
`day( 35648 )`	retorna 6, por que 35648 = 1997-08-06

Exemplo 1 – conjunto de dados DateFormat (script)

Script de carregamento e resultados

Visão geral

Abra o Editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados de datas chamado `Master_Calendar`. A variável do sistema `DateFormat` é definida como DD/MM/AAAA.
- Um carregamento anterior que cria um campo adicional, chamado `day_of_month`, usando a função `day ()`.
- Um campo adicional, chamado `long_date`, usando a função `date()` para expressar o nome completo do mês.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `long_date`
- `day_of_month`

Tabela de resultados

data	long_date	day_of_month
03/11/2022	11-Março- 2022	11
03/12/2022	12-Março- 2022	12
03/13/2022	13-Março- 2022	13
03/14/2022	14-Março- 2022	14
03/15/2022	15-Março- 2022	15
03/16/2022	16-Março- 2022	16
03/17/2022	17-Março- 2022	17
03/18/2022	18-Março- 2022	18
03/19/2022	19-Março- 2022	19

data	long_date	day_of_month
03/20/2022	20-Março- 2022	20
03/21/2022	21-Março- 2022	21

O dia do mês é avaliado corretamente pela função `day()` no script.

Exemplo 2 – datas ANSI (script)

Script de carregamento e resultados

Visão geral

Abra o Editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados de datas chamado `Master_Calendar`. A variável de sistema DateFormat DD/MM/AAAA é usada. No entanto, as datas incluídas no conjunto de dados estão no formato de data padrão ANSI.
- Um carregamento anterior que cria um campo adicional, chamado `day_of_month`, usando a função `date()`.
- Um campo adicional, chamado `long_date`, usando a função `date()` para expressar a data com o nome completo do mês.

Script de carregamento

```
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
 ];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `long_date`
- `day_of_month`

Tabela de resultados

data	long_date	day_of_month
03/11/2022	11-Março- 2022	11
03/12/2022	12-Março- 2022	12
03/13/2022	13-Março- 2022	13
03/14/2022	14-Março- 2022	14
03/15/2022	15-Março- 2022	15
03/16/2022	16-Março- 2022	16
03/17/2022	17-Março- 2022	17
03/18/2022	18-Março- 2022	18
03/19/2022	19-Março- 2022	19
03/20/2022	20-Março- 2022	20
03/21/2022	21-Março- 2022	21

O dia do mês é avaliado corretamente pela função `day()` no script.

Exemplo 3 – Datas não formatadas (script)

Script de carregamento e resultados

Visão geral

Abra o Editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados de datas chamado `Master_Calendar`. A variável de sistema `DateFormat` DD/MM/AAAA é usada.
- Um carregamento anterior que cria um campo adicional, chamado `day_of_month`, usando a função `day()`.
- A data original não formatada, chamada `unformatted_date`.
- Um campo adicional, chamado `long_date`, usando o `date()` é usado para converter a data numérica em um campo de data formatado.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `unformatted_date`
- `long_date`
- `day_of_month`

Tabela de resultados

unformatted_date	long_date	day_of_month
44868	03-Novembro- 2022	3
44898	03-Dezembro- 2022	3
44928	02-Janeiro- 2023	2
44958	01-Fevereiro- 2023	1
44988	03-Março- 2023	3
45008	23-Março- 2023	23
45018	02-Abril- 2023	2
45038	22-Abril- 2023	22

unformatted_date	long_date	day_of_month
45048	02-Maio- 2023	2
45068	22-Maio- 2023	22
45078	01-Junho- 2023	1

O dia do mês é avaliado corretamente pela função `day()` no script.

Exemplo 4 – Cálculo do mês de vencimento (gráfico)

Script de carregamento e expressão de gráfico

Visão geral

Abra o Editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados de pedidos feitos em março chamado `Orders`. A tabela contém três campos:
  - id
  - order_date
  - amount

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão:`order_date`.

Para calcular a data de entrega, crie esta medida `=day(order_date+5)`.

Tabela de resultados

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

A função `day()` determina corretamente que um pedido feito no dia 11 de março será entregue no dia 16 com base em um prazo de entrega de 5 dias.

dayend

Esta função retorna um valor correspondente a uma data/hora com o milissegundo final do dia contido em **time**. O formato de saída padrão será o **TimestampFormat** definido no script.

Sintaxe:

```
DayEnd(time[, [period_no[, day_start]])
```

Quando usar

A função `dayend()` é comumente usada como parte de uma expressão quando o usuário deseja que o cálculo use a fração do dia que ainda não ocorreu. Por exemplo, para calcular as despesas totais que ainda serão acumuladas ao longo do dia.

Tipo de dados de retorno: dual

Argumentos

Argumento	Descrição
time	A data/hora para avaliar.
period_no	**period_no** é um inteiro, ou uma expressão que resolve um inteiro, em que o valor 0 indica o dia que contém o **time**. Valores negativos em **period_no** indicam dias precedentes e valores positivos indicam dias sucessivos.
day_start	Para especificar que os dias não comecem à meia-noite, indique uma diferença como uma fração de um dia em **day_start**. Por exemplo, 0,125 para representar 3:00 a.m. Em outras palavras, para criar o deslocamento, divida a hora de início por 24 horas. Por exemplo, para um dia começar às 7:00 a.m., use a fração 7/24.

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplos de funções

Exemplo	Resultado
dayend('01/25/2013 16:45:00')	Retorna 01/25/2013 23:59:59. PM
dayend('01/25/2013 16:45:00', -1)	Retorna 24/01/2013 23:59:59. PM
dayend('01/25/2013 16:45:00', 0, 0.5)	Retorna 01/26/2013 11:59:59. PM

Exemplo 1 – Script básico

Script de carregamento e resultados

Visão geral

Abra o Editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo uma lista de datas é carregado em uma tabela chamada "Calendar".
- A variável de sistema `DateFormat` padrão `(MM/DD/YYYY)`.
- Um carregamento anterior para criar um campo adicional, '`EOD_timestamp`', usando a função `dayend ()`.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `EOD_timestamp`

Tabela de resultados

data	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
03/15/2022 10:06:54 AM	3/15/2022 11:59:59 PM

data	EOD_timestamp
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

Como você pode ver na tabela acima, o carimbo de data/hora do fim do dia é gerado para cada data em nosso conjunto de dados. O carimbo de data/hora está no formato da variável do sistema `TimestampFormat M/D/YYYY h:mm:ss[.fff] TT`.

Exemplo 2 – period_no

Script de carregamento e resultados

Visão geral

Abra o Editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia.

Você carregará um conjunto de dados contendo reservas de serviço em uma tabela chamada "Services".

O conjunto de dados inclui os seguintes campos:

- `service_id`
- `service_date`
- `amount`

Você criará dois novos campos na tabela:

- `deposit_due_date`: A data em que o depósito deve ser recebido. Este é o final do dia, três dias antes de `service_date`.
- `final_payment_due_date`: A data em que o pagamento final deve ser recebido. Este é o final do dia, sete dias após `service_date`.

Os dois campos acima são criados em uma carga anterior usando a função `dayend()` e fornecem os dois primeiros parâmetros: `time` e `period_no`.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- service_date
- deposit_due_date
- final_payment_due_date

Tabela de resultados

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

Os valores dos novos campos estão em `TimestampFormat M/D/YYYY h:mm:ss[.fff] TT`. Como a função `dayend()` foi usada, os valores de carimbo de data/hora são todos os últimos milissegundos do dia.

Os valores da data de vencimento do depósito são três dias antes da data do serviço, pois o segundo argumento transmitido na função `dayend()` é negativo.

Os valores da data de vencimento do pagamento final são sete dias após a data do serviço, pois o segundo argumento transmitido na função `dayend()` é positivo.

Exemplo 3 – script day_start

Script de carregamento e resultados

Visão geral

Abra Editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia.

O conjunto de dados e o cenário usados neste exemplo são os mesmos do exemplo anterior.

Como no exemplo anterior, você criará dois novos campos:

- `deposit_due_date`: A data em que o depósito deve ser recebido. Este é o final do dia, três dias antes de `service_date`.
- `final_payment_due_date`: A data em que o pagamento final deve ser recebido. Este é o final do dia, sete dias após `service_date`.

No entanto, sua empresa gostaria de operar de acordo com uma política em que o dia útil começa às 17h e termina às 17h do dia seguinte. Ela pode então monitorar as transações que ocorrem nessas horas de trabalho.

Para atender a esses requisitos, os dois campos acima são criados em um carregamento anterior usando a função `dayend()` e utilizam todos os três argumentos: `time`, `period_no` e `day_start`.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `service_date`
- `deposit_due_date`
- `final_payment_due_date`

Tabela de resultados

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

Embora as datas permaneçam as mesmas do Exemplo 2, elas agora têm um carimbo de data/hora do último milissegundo antes das 17h, pois o valor do terceiro argumento, `day_start`, passado para a função `dayend()` é 17/24.

Exemplo 4 – Exemplo de gráfico

Script de carregamento e expressão de gráfico

Visão geral

Abra o Editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia.

O conjunto de dados e o cenário usados neste exemplo são os mesmos dos dois exemplos anteriores. Sua empresa gostaria de operar de acordo com uma política em que o dia útil começa às 17h e termina às 17h do dia seguinte.

Como no exemplo anterior, você criará dois novos campos:

- `deposit_due_date`: A data em que o depósito deve ser recebido. Este é o final do dia, três dias antes de `service_date`.
- `final_payment_due_date`: A data em que o pagamento final deve ser recebido. Este é o final do dia, sete dias após `service_date`.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão:

`service_date`.

Para criar o campo `deposit_due_date`, crie esta medida.

```
=dayend(service_date,-3,17/24).
```

Em seguida, para criar o campo `final_payment_due_date`, crie esta medida:

```
=dayend(service_date,7,17/24).
```

Tabela de resultados

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

Os valores dos novos campos estão em `TimestampFormat M/D/YYYY h:mm:ss[.fff] TT`. Como a função `dayend()` foi usada, os valores de carimbo de data/hora são todos os últimos milissegundos do dia.

Os valores da data de vencimento do pagamento são três dias antes da data do serviço, pois o segundo argumento transmitido na função `dayend()` é negativo.

Os valores da data de vencimento do pagamento final são sete dias após a data do serviço, pois o segundo argumento transmitido na função `dayend()` é positivo.

As datas têm um carimbo de data/hora do último milissegundo antes das 17h, pois o valor do terceiro argumento, `day_start`, transmitido para a função `dayend()` é 17/24.

Argumentos

Argumento	Descrição
time	A data/hora para avaliar.
period_no	**period_no** é um inteiro, ou uma expressão que resolve um inteiro, em que o valor 0 indica o dia que contém o **time**. Valores negativos em **period_no** indicam dias precedentes e valores positivos indicam dias sucessivos.
day_start	Para especificar que os dias não comecem à meia-noite, indique uma diferença como uma fração de um dia em **day_start**. Por exemplo, 0,125 para representar 3:00 a.m.

daylightsaving

Retorna os ajustes atuais do horário de verão conforme definido no Windows.

Sintaxe:

```
DaylightSaving( )
```

Tipo de dados de retorno: dual

Exemplo:

```
daylightsaving( )
```

dayname

Esta função retorna um valor que mostra a data com um valor numérico subjacente que corresponde a um carimbo de hora do primeiro milissegundo do dia que contém o **time**.

Sintaxe:

```
DayName(time[, period_no [, day_start]])
```

Tipo de dados de retorno: dual

Argumentos:

Argumentos

Argumento	Descrição
time	A data/hora para avaliar.
period_no	**period_no** é um inteiro, ou uma expressão que resolve um inteiro, em que o valor 0 indica o dia que contém o **time**. Valores negativos em **period_no** indicam dias precedentes e valores positivos indicam dias sucessivos.
day_start	Para especificar que os dias não comecem à meia-noite, indique uma diferença como uma fração de um dia em **day_start**. Por exemplo, 0,125 para representar 3:00 a.m.

Exemplos e resultados:
Estes exemplos usam o formato de data **DD/MM/YYYY**. O formato de data é especificado no comando **SET DateFormat** na parte superior do seu script de carga de dados. Altere o formato nos exemplos para atender às suas necessidades.

Exemplos de scripts

Exemplo	Resultado
`dayname('25/01/2013 16:45:00')`	Retorna 25/01/2013.
`dayname('25/01/2013 16:45:00', -1)`	Retorna 24/01/2013.
`dayname('25/01/2013 16:45:00', 0, 0.5 )`	Returna 25/01/2013. A exibição de toda a data/hora mostra o valor numérico subjacente correspondente a '25/01/2013 12:00:00.000.

Exemplo:

Adicione o script de exemplo ao seu aplicativo e execute-o. Para ver o resultado, adicione os campos listados na coluna de resultados a uma pasta no seu aplicativo.

Neste exemplo, o nome do dia é criado a partir da data/hora que marca o início do dia depois de cada data da fatura na tabela.

```
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;
```

A tabela resultante contém as datas originais e uma coluna com o valor de retorno da função dayname(). É possível exibir toda a data/hora ao especificar a formatação no painel de propriedades.

Tabela de resultados

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

Esta função calcula o número do dia do trimestre em que um timestamp cai. Essa função é usada ao criar um calendário mestre.

Sintaxe:

```
DayNumberOfQuarter(timestamp[,start_month])
```

Tipo de dados de retorno: inteiro

Argumentos

Argumento	Descrição
timestamp	A data ou o carimbo de data/hora a ser avaliado.
start_month	Ao especificar um **start_month** entre 2 e 12 (1, se omitido), o início do ano pode avançar para o primeiro dia de qualquer mês. Por exemplo, para trabalhar com um ano fiscal que inicia em 1º de março, especifique **start_month** = 3.

Estes exemplos usam o formato de data **DD/MM/YYYY**. O formato de data é especificado no comando **SET DateFormat** na parte superior do seu script de carga de dados. Altere o formato nos exemplos para atender às suas necessidades.

Exemplos de funções

Exemplo	Resultado
`DayNumberOfQuarter('12/09/2014')`	Retorna 74, o número do dia do trimestre atual.
`DayNumberOfQuarter ('12/09/2014',3)`	Retorna 12, o número do dia do trimestre atual. Neste caso, o primeiro trimestre começa em março (pois o start_month está especificado como 3). Isto significa que o trimestre atual é o terceiro semestre, que começou dia 1 de setembro.

Exemplo 1 – Início do ano em janeiro (script)

Script de carregamento e resultados

Visão geral

Abra o Editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados simples contendo uma lista de datas, que é carregada em uma tabela chamada `Calendar`. A variável de sistema `DateFormat` padrão MM/DD/AAAA é usada.
- Um carregamento anterior que cria um campo adicional, chamado `DayNrQtr`, usando a função `DayNumberOfQuarter()`.

Além da data, nenhum parâmetro adicional é fornecido para a função.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `daynrqtr`

Tabela de resultados

data	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

O primeiro dia do ano é 1º de janeiro porque um segundo argumento não foi transmitido para a função `DayNumberOfQuarter()`.

1º de janeiro é o 1º dia do trimestre, enquanto 1º de fevereiro é o 32º dia do trimestre. 31 de março é o 91º e último dia do trimestre, enquanto o 1º de abril é o 1º dia do 2º trimestre.

Exemplo 2 – Início do ano em fevereiro (script)

Script de carregamento e resultados

Visão geral

Abra o Editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- O mesmo conjunto de dados do primeiro exemplo.
- A variável de sistema `DateFormat` padrão MM/DD/AAAA é usada.
- Um argumento `start_month` que começa em 1º de fevereiro. Isso define o exercício como 1º de fevereiro.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `daynrqtr`

Tabela de resultados

data	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

O primeiro dia do ano é 1º de fevereiro, pois o segundo argumento transmitido para a função `DayNumberOfQuarter()` foi 2.

O primeiro trimestre do ano opera entre fevereiro e abril, enquanto o quarto trimestre opera entre novembro e janeiro. Isso é mostrado na tabela de resultados, onde 1º de fevereiro é o 1º dia do trimestre, enquanto 31 de janeiro é o 92º e último dia do trimestre.

Exemplo 3 – Início do ano em janeiro (gráfico)

Script de carregamento e expressão de gráfico

Visão geral

Abra o Editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- O mesmo conjunto de dados do primeiro exemplo.
- A variável de sistema `DateFormat` padrão MM/DD/AAAA é usada.

No entanto, neste exemplo, o conjunto de dados inalterado é carregado no aplicativo. O valor do dia do trimestre é calculado por meio de uma medida em um objeto de gráfico.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão: `date`.

Crie a seguinte medida:

```
=daynumberofquarter(date)
```

Tabela de resultados

data	=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

O primeiro dia do ano é 1º de janeiro, pois um segundo argumento não foi transmitido para a função `DayNumberOfQuarter()`.

1º de janeiro é o 1º dia do trimestre, enquanto 1º de fevereiro é o 32º dia do trimestre. 31 de março é o 91º e último dia do trimestre, enquanto o 1º de abril é o 1º dia do 2º trimestre.

Exemplo 4 – Início do ano em fevereiro (gráfico)

Script de carregamento e expressão de gráfico

Visão geral

Abra o Editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- O mesmo conjunto de dados do primeiro exemplo.
- A variável de sistema `DateFormat` padrão MM/DD/AAAA é usada.
- O exercício vai de 1º de fevereiro a 31 de janeiro.

No entanto, neste exemplo, o conjunto de dados inalterado é carregado no aplicativo. O valor do dia do trimestre é calculado por meio de uma medida em um objeto de gráfico.

Script de carregamento

```
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
];
```

Objeto de gráfico

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão: `date`.

Crie a seguinte medida:

```
=daynumberofquarter(date,2)
```

Resultados

Tabela de resultados

data	=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

O primeiro dia do ano é 1º de janeiro, pois o segundo argumento transmitido para a função `DayNumberOfQuarter()` foi 2.

O primeiro trimestre do ano opera entre fevereiro e abril, enquanto o quarto trimestre opera entre novembro e janeiro. Isso é evidenciado na tabela de resultados, onde 1º de fevereiro é o 1º dia do trimestre, enquanto 31 de janeiro é o 92º e último dia do trimestre.

daynumberofyear

Essa função calcula o número do dia do ano que estiver em um carimbo de data/hora. O cálculo é feito a partir do primeiro milissegundo do primeiro dia do ano, mas o primeiro mês pode ser deslocado.

Sintaxe:

```
DayNumberOfYear(timestamp[,start_month])
```

Tipo de dados de retorno: inteiro

Argumentos

Argumento	Descrição
timestamp	A data ou o carimbo de data/hora a ser avaliado.
start_month	Ao especificar um **start_month** entre 2 e 12 (1, se omitido), o início do ano pode avançar para o primeiro dia de qualquer mês. Por exemplo, para trabalhar com um ano fiscal que inicia em 1º de março, especifique **start_month** = 3.

Estes exemplos usam o formato de data **DD/MM/YYYY**. O formato de data é especificado no comando **SET DateFormat** na parte superior do seu script de carga de dados. Altere o formato nos exemplos para atender às suas necessidades.

Exemplos de funções

Exemplo	Resultado
`DayNumberOfYear( '12/09/2014' )`	Retorna 256, o número do dia contado a partir do primeiro do ano.
`DayNumberOfYear( '12/09/2014',3 )`	Retorna 196, o número do dia contado a partir de 1 de março.

Exemplo 1 – Início do ano em janeiro (script)

Script de carregamento e resultados

Visão geral

Abra o Editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados simples contendo uma lista de datas, que é carregada em uma tabela chamada `Calendar`. A variável de sistema `DateFormat` padrão MM/DD/AAAA é usada.
- Um carregamento anterior que cria um campo adicional, chamado `daynryear`, usando a função `DayNumberOfYear()`.

Além da data, nenhum parâmetro adicional é fornecido para a função.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `daynryear`

Tabela de resultados

data	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

O primeiro dia do ano é 1º de janeiro, pois um segundo argumento não foi transmitido para a função `DayNumberOfYear()`.

1º de janeiro é o 1º dia do trimestre, enquanto 1º de fevereiro é o 32º dia do ano. O dia 30 de junho é o 182º, enquanto 31 de dezembro é o 366º e último dia do ano.

Exemplo 2 – Início do ano de novembro (script)

Script de carregamento e resultados

Visão geral

Abra o Editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- O mesmo conjunto de dados do primeiro exemplo.
- A variável de sistema `DateFormat` padrão MM/DD/AAAA é usada.
- Um argumento `start_month` que começa em 1º de novembro. Isso define o exercício como 1º de novembro.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `daynryear`

Tabela de resultados

data	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

O primeiro dia do ano é 1º de novembro, pois o segundo argumento transmitido para a função `DayNumberOfYear()` foi 11.

1º de janeiro é o 1º dia do trimestre, enquanto 1º de fevereiro é o 32º dia do ano. O dia 30 de junho é o 182º, enquanto 31 de dezembro é o 366º e último dia do ano.

Exemplo 3 – Início do ano em janeiro (gráfico)

Script de carregamento e expressão de gráfico

Visão geral

Abra o Editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- O mesmo conjunto de dados do primeiro exemplo.
- A variável de sistema `DateFormat` padrão MM/DD/AAAA é usada.

No entanto, neste exemplo, o conjunto de dados inalterado é carregado no aplicativo. O valor do dia do trimestre é calculado por meio de uma medida em um objeto de gráfico.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão: `date`.

Crie a seguinte medida:

```
=daynumberofyear(date)
```

Tabela de resultados

data	=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

O primeiro dia do ano é 1º de janeiro, pois um segundo argumento não foi transmitido para a função `DayNumberOfYear()`.

1º de janeiro é o 1º dia do ano, enquanto 1º de fevereiro é o 32º dia do ano. O dia 30 de junho é o 182º, enquanto 31 de dezembro é o 366º e último dia do ano.

Exemplo 4 – Início do ano em novembro (gráfico)

Script de carregamento e expressão de gráfico

Visão geral

Abra o Editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- O mesmo conjunto de dados do primeiro exemplo.
- A variável de sistema `DateFormat` padrão MM/DD/AAAA é usada.
- O exercício vai de 1º de novembro a 31 de outubro.

No entanto, neste exemplo, o conjunto de dados inalterado é carregado no aplicativo. O valor do dia do ano é calculado por meio de uma medida em um objeto de gráfico.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão: `date`.

Crie a seguinte medida:

```
=daynumberofyear(date)
```

Tabela de resultados

data	=daynumberofyear(date,11)
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

data	=daynumberofyear(date,11)
10/31/2022	366
11/01/2022	1
12/31/2022	61

O primeiro dia do ano é 1º de novembro, pois o segundo argumento transmitido para a função `DayNumberOfYear()` foi 11.

O exercício opera entre novembro e outubro. Isso é mostrado na tabela de resultados, onde 1º de novembro é o 1º dia do ano, enquanto 31 de outubro é o 366º e último dia do ano.

daystart

Esta função retorna um valor correspondente a uma data/hora com o primeiro milissegundo do dia contido no argumento **time**. O formato de saída padrão será o **TimestampFormat** definido no script.

Sintaxe:

```
DayStart(time[, [period_no[, day_start]])
```

Tipo de dados de retorno: dual

Argumentos

Argumento	Descrição
time	A data/hora para avaliar.
period_no	**period_no** é um inteiro, ou uma expressão que resolve um inteiro, em que o valor 0 indica o dia que contém o **time**. Valores negativos em **period_no** indicam dias precedentes e valores positivos indicam dias sucessivos.
day_start	Para especificar que os dias não comecem à meia-noite, indique uma diferença como uma fração de um dia em **day_start**. Por exemplo, 0,125 para representar 3:00 a.m. Em outras palavras, para criar o deslocamento, divida a hora de início por 24 horas. Por exemplo, para um dia começar às 7:00 a.m., use a fração 7/24.

Quando usar

A função `daystart()` é comumente usada como parte de uma expressão quando o usuário deseja que o cálculo use a fração do dia decorrido até o momento. Por exemplo, ela pode ser usada para calcular o total de salários ganhos pelos funcionários no dia até o momento.

Esses exemplos usam o formato de carimbo de data/hora `'M/D/YYYY h:mm:ss[.fff] TT'`. O formato do carimbo de data/hora é especificado na instrução `SET TimeStamp`, na parte superior do script de carregamento de dados. Altere o formato nos exemplos para atender às suas necessidades.

Exemplos de funções

Exemplo	Resultado
`daystart('01/25/2013 4:45:00 PM')`	Retorna 1/25/2013 12:00:00 AM.
`daystart('1/25/2013 4:45:00 PM', -1)`	Retorna 1/24/2013 12:00:00 AM.
`daystart('1/25/2013 16:45:00',0,0.5 )`	Retorna 1/25/2013 12:00:00 PM.

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplo 1 - Exemplo simples

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados simples contendo uma lista de datas, que é carregada em uma tabela chamada `Calendar`.
- A `variável` de sistema padrão TimeStampFormat (`(M/D/YYYY h:mm:ss[.fff] TT`) é usada.
- Um carregamento anterior que cria um campo adicional, denominado `SOD_timestamp`, usando a função `daystart()`.

Além da data, nenhum parâmetro adicional é fornecido para a função.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `SOD_timestamp`

Tabela de resultados

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
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

Como você pode ver na tabela acima, o carimbo de data/hora do fim do dia é gerado para cada data em nosso conjunto de dados. O carimbo de data/hora está no formato da variável do sistema `TimestampFormat M/D/YYYY h:mm:ss[.fff] TT`.

Exemplo 2: period_no

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo multas de estacionamento, que é carregado em uma tabela denominada `Fines`. O conjunto de dados inclui os seguintes campos:
    - `id`
    - `due_date`
    - `number_plate`
    - `amount`
- Um carregamento anterior usando a função `daystart()` e fornecendo todos os três parâmetros: `time`, `period_no` e `day_start`. Esse carregamento anterior cria os dois novos campos de data a seguir:
    - Um campo de data `early_repayment_period`, começando sete dias antes do vencimento do pagamento.
    - Um campo de data `late_penalty_period`, começando 14 dias após o vencimento do pagamento.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- due_date
- early_repayment_period
- late_penalty_period

Tabela de resultados

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

Os valores dos novos campos estão em `TimestampFormat M/DD/YYYY tt`. Como a função `daystart()` foi usada, os valores de carimbo de data/hora são todos os primeiros milissegundos do dia.

Os valores do período de reembolso antecipado são sete dias antes da data de vencimento, como resultado de o segundo argumento transmitido na função `daystart()` ser negativo.

Os valores do período de reembolso tardio são 14 dias após a data de vencimento, como resultado de o segundo argumento transmitido na função `daystart()` ser positivo.

Exemplo 3: day_start

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- O mesmo conjunto de dados e cenário do exemplo anterior.
- O mesmo carregamento anterior que o exemplo anterior.

Neste exemplo, definimos o dia útil para começar e terminar às 7:00 AM todos os dias.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- due_date
- early_repayment_period
- late_penalty_period

Tabela de resultados

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

As datas agora têm um carimbo de data/hora de 7:00 AM porque o valor do argumento day_start que foi transmitido para a função daystart() era 7/24. Isso define o início do dia como 7:00 AM.

Como o campo due_date não tem um carimbo de data/hora, ele é tratado como 12:00 AM, o que, portanto, ainda faz parte do dia anterior, já que os dias começam e terminam às 7:00 AM. Portanto, o período de reembolso antecipado de uma multa devida em 11 de fevereiro começa em 3 de fevereiro às 7:00 AM.

Exemplo 4: Exemplo de objeto de gráfico

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia.

Este exemplo usa o mesmo conjunto de dados e cenário do exemplo anterior.

No entanto, somente a tabela Fines original é carregada no aplicativo, com os dois valores adicionais de datas de vencimento sendo calculados em um objeto de gráfico.

Script de carregamento

```
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
];
```

Resultados

Faça o seguinte:

1. Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão: due_date.
2. Para criar o campo early_repayment_period, crie a seguinte medida: =daystart(due_date,-7,7/24)
3. Para criar o campo late_penalty_period, crie a seguinte medida: =daystart(due_date,14,7/24)

Tabela de resultados

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

Os valores dos novos campos estão em `TimestampFormat M/D/YYYY h:mm:ss[.fff] TT`. Como a função `daystart()` foi usada, os valores de carimbo de data/hora correspondem ao primeiro milissegundo do dia.

Os valores do período de reembolso antecipado são sete dias antes da data de vencimento, já que o segundo argumento transmitido na função `daystart()` era negativo.

Os valores do período de reembolso tardio são 14 dias após a data de vencimento, já que o segundo argumento transmitido na função `daystart()` era positivo.

As datas têm um carimbo de data/hora de 7:00 AM, pois o valor do terceiro argumento transmitido para a função `daystart()`, `day_start`, era 7/24.

firstworkdate

A função **firstworkdate** retorna a última data inicial para obter o **no_of_workdays** (segunda-sexta) com término não posterior à **end_date** levando em conta os feriados opcionalmente listados. **end_date** e **holiday** devem ser datas ou carimbos de data/hora válidos.

Sintaxe:

```
firstworkdate(end_date, no_of_workdays {, holiday} )
```

Tipo de dados de retorno: inteiro

Argumentos:

Argumentos

Argumento	Descrição
end_date	Data/hora da data de término para avaliar.
no_of_ workdays	Número de dias úteis para alcançar.

Argumento	Descrição
holiday	Períodos de feriados a serem excluídos dos dias de trabalho. Um feriado é declarado como uma data constante de string. Você pode especificar várias datas de feriados, separadas por vírgulas. **Exemplo:** `'12/25/2013', '12/26/2013', '12/31/2013', '01/01/2014'`

Exemplos e resultados:
Estes exemplos usam o formato de data **DD/MM/YYYY**. O formato de data é especificado no comando **SET DateFormat** na parte superior do seu script de carga de dados. Altere o formato nos exemplos para atender às suas necessidades.

Exemplos de scripts

Exemplo	Resultado
`firstworkdate ('29/12/2014', 9)`	Retorna '17/12/2014.
`firstworkdate ('29/12/2014', 9, '25/12/2014', '26/12/2014')`	Retorna 15/12/2014, pois um feriado de dois dias é considerado.

Exemplo:

Adicione o script de exemplo ao seu aplicativo e execute-o. Para ver o resultado, adicione os campos listados na coluna de resultados a uma pasta no seu aplicativo.

```
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;
```

A tabela resultante mostra os valores retornados de FirstWorkDate para cada um dos registros na tabela.

Tabela de resultados

InvID	EndDate	StartDate
1	28/03/2015	13/10/2014
2	10/12/2015	26/06/2015

InvID	EndDate	StartDate
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

Essa função retorna o Greenwich Mean Time atual, derivado das configurações regionais. A função retorna valores no formato da variável do sistema `TimestampFormat`.

Sempre que o aplicativo for carregado, qualquer tabela, variável ou objeto de gráfico de script de carregamento que use a função `GMT` será ajustado para o horário médio de Greenwich atual mais recente, conforme derivado do relógio do sistema.

Sintaxe:

```
GMT( )
```

Tipo de dados de retorno: dual

Esses exemplos usam o formato de carimbo de data/hora `M/D/YYYY h:mm:ss[.fff] TT`. O formato de data é especificado no comando `SET TimestampFormat` na parte superior do seu script de carregamento de dados. Altere o formato nos exemplos para atender às suas necessidades.

Exemplos de funções

Exemplo	Resultado
`GMT()`	3/28/2022 2:47:36 PM

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplo 1: Variável (script)

Script de carregamento e resultados

Visão geral

Abra o editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia. Este exemplo definirá o Horário médio de Greenwich atual como uma variável no script de carregamento usando a função GMT.

Script de carregamento

```
LET vGMT = GMT();
```

Resultados

Carregue os dados e crie uma pasta. Crie uma caixa de texto usando o objeto de gráfico de **Texto e imagem**.

Adicione essa medida à caixa de texto:

```
=vGMT
```

A caixa de texto deve conter uma linha de texto com data e hora, semelhante à mostrada abaixo:

```
3/28/2022 2:47:36 PM
```

Exemplo 2: Início do ano de novembro (script)

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo livros da biblioteca vencidos, que é carregado em uma tabela denominada Overdue. A variável de sistema DateFormat padrão MM/DD/AAAA é usada.
- A criação de um novo campo denominado days_overdue, que calcula com quantos dias de atraso cada livro está.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- due_date
- book_id
- days_overdue

Tabela de resultados

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
07/26/2021	537	249
10/31/2021	275	152
11/01/2021	455	151
12/31/2021	46	91

Os valores no campo days_overdue são calculados com base na diferença entre o Horário médio de Greenwich atual, usando a função GMT(), e a data de vencimento original. Para calcular somente os dias, os resultados são arredondados para o número inteiro mais próximo usando a função Floor().

Exemplo 3: Objeto de gráfico (gráfico)

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia. O script de carregamento contém o mesmo conjunto de dados do exemplo anterior. A variável de sistema `DateFormat` padrão MM/DD/AAAA é usada.

No entanto, neste exemplo, o conjunto de dados inalterado é carregado no aplicativo. O valor do número de dias em atraso é calculado por meio de uma medida em um objeto de gráfico.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `due_date`
- `book_id`

Crie a seguinte medida:

```
=Floor(GMT() - due_date)
```

Tabela de resultados

due_date	book_id	=Floor(GMT()-due_date)
01/01/2021	4	455

due_date	book_id	=Floor(GMT()-due_date)
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

Os valores no campo `days_overdue` são calculados com base na diferença entre o Horário médio de Greenwich atual, usando a função `GMT()`, e a data de vencimento original. Para calcular somente os dias, os resultados são arredondados para o número inteiro mais próximo usando a função `Floor()`.

hour

Esta função retorna um número inteiro que representa a hora em que a fração da **expression** é interpretada como uma hora, de acordo com a interpretação numérica padrão.

Sintaxe:

```
hour(expression)
```

Tipo de dados de retorno: inteiro

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplos de funções

Exemplo	Resultado
`hour( '09:14:36' )`	A string de texto fornecida é implicitamente convertida em um carimbo de data/hora, pois corresponde ao formato de carimbo de data/hora definido na variável TimestampFormat. A expressão retorna 9.
`hour( '0.5555' )`	A expressão retorna 13 (porque 0,5555 = 13:19:55).

Exemplo 1 – Variável (script)

Script de carregamento e resultados

Visão geral

Abra o Editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo transações por carimbo de data/hora.
- A variável de sistema `TimeStamp` padrão `(M/D/YYYY h:mm:ss[.fff] TT)`

Crie um campo, 'hour', calculando quando as compras ocorreram.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `hour`

Tabela de resultados

data	hora
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

Os valores no campo de hora são criados usando a função `hour()` função e transferindo a data como a expressão no comando de carregamento anterior.

Exemplo 2 – Objeto de gráfico (gráfico)

Script de carregamento e expressão de gráfico

Visão geral

Abra o Editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- O mesmo conjunto de dados do primeiro exemplo.
- A variável de sistema `TimeStamp` padrão `(M/D/YYYY h:mm:ss[.fff] TT)`.

No entanto, neste exemplo, o conjunto de dados inalterado é carregado no aplicativo. Os valores de 'hour' são calculados por meio de uma medida em um objeto de gráfico.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão: `date`.

Para calcular "`hour`", crie a seguinte medida:

```
=hour(date)
```

Tabela de resultados

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
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

Os valores para "`hour`" são criados usando a função `hour()` e transmitindo a data como a expressão em uma medida para o objeto de gráfico.

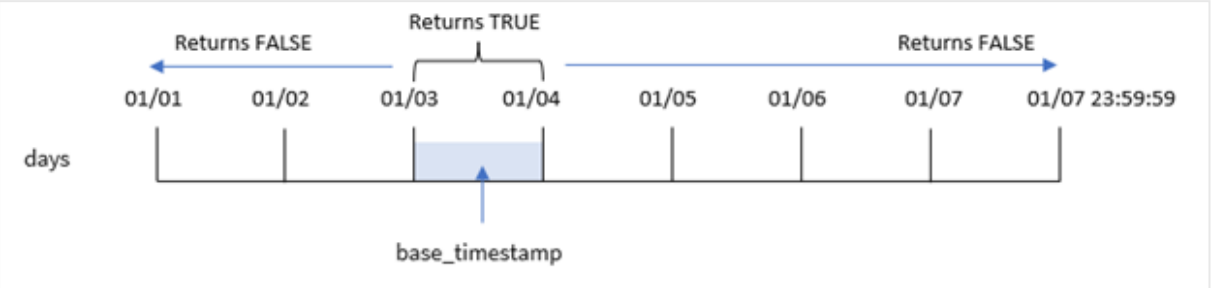
inday

Esta função retorna True se **timestamp** estiver dentro do dia que contém **base_timestamp**.

Sintaxe:

```
InDay (timestamp, base_timestamp, period_no[, day_start])
```

Diagrama da função inday



A função `inday()` usa o argumento `base_timestamp` para identificar em qual dia cai o carimbo de data/hora. A hora de início do dia é, por padrão, meia-noite, mas você pode alterá-la usando o argumento `day_start` da função `inday()`. Assim que esse dia for definido, a função retornará resultados booleanos ao comparar os valores de carimbo de data/hora prescritos com aquele dia.

Quando usar

A função `inday()` retorna um resultado booleano. Normalmente, esse tipo de função será usado como uma condição em um `if expression`. Isso retorna uma agregação ou cálculo dependente de uma data avaliada ter ocorrido no dia do carimbo de data/hora em questão.

Por exemplo, a função `inday()` pode ser usada para identificar todos os equipamentos fabricados em um determinado dia.

Tipo de dados de retorno: Booleano

No Qlik Sense, o valor booleano “true” é representado por -1, e o valor falso é representado por 0.

Argumentos

Argumento	Descrição
`timestamp`	A data e hora que você deseja comparar com `base_timestamp`.
`base_timestamp`	Data e hora que são usadas para avaliar o carimbo de data/hora.
`period_no`	O dia pode ser deslocado por `period_no`. `period_no` é um inteiro, em que o valor 0 indica o dia que contém `base_timestamp`. Valores negativos em `period_no` indicam dias precedentes e valores positivos indicam dias sucessivos.
`day_start`	Se você desejar trabalhar com dias que não comecem à meia-noite, indique um deslocamento em fração de um dia em `day_start`, por exemplo, 0,125 para indicar 3h da manhã.

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplos de funções

Exemplo	**Resultado**
`inday` ('01/12/2006 12:23:00 PM', '01/12/2006 12:00:00 AM', 0)	Retorna True
`inday` ('01/12/2006 12:23:00 PM', '01/13/2006 12:00:00 AM', 0)	Retorna False
`inday` ('01/12/2006 12:23:00 PM', '01/12/2006 12:00:00 AM', -1)	Retorna False
`inday` ('01/11/2006 12:23:00 PM', '01/12/2006 12:00:00 AM', -1)	Retorna True
`inday` ('01/12/2006 12:23:00 PM', '01/12/2006 12:00:00 AM', 0, 0.5)	Retorna False
`inday` ('01/12/2006 11:23:00 AM', '01/12/2006 12:00:00 AM', 0, 0.5)	Retorna True

Exemplo 1 – Instrução de carregamento (script)

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo transações por carimbo de data/hora que é carregado em uma tabela denominada `Transactions`.
- Um campo de data que é fornecido na `variável` do sistema TimeStamp, no formato `(M/D/YYYY h:mm:ss[.fff] TT)`.
- Um carregamento anterior que contém a função `inday()` que é definida como o campo `in_day`.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `in_day`

Tabela de resultados

data	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

O campo `in_day` é criado na instrução de carregamento anterior usando a função `inday()` e transmitindo o campo de data, um carimbo de data/hora codificado para 5 de janeiro e um `period_no` de `0` como os argumentos da função.

Exemplo 2 – period_no

Script de carregamento e resultados

Visão geral

O script de carregamento usa o mesmo conjunto de dados e cenário que foram usados no primeiro exemplo.

No entanto, neste exemplo, a tarefa é calcular se a data da transação ocorreu dois dias antes de 5 de janeiro.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `in_day`

Tabela de resultados

data	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

data	in_day
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

Nesse caso, como um `period_no` de `-2` é usado como argumento de deslocamento na função `inday()`, essa determina se cada data de transação ocorreu em 3 de janeiro. Isso pode ser verificado na tabela de saída, onde uma transação retorna um resultado booleano de TRUE.

Exemplo 3 – day_start

Script de carregamento e resultados

Visão geral

O script de carregamento usa o mesmo conjunto de dados e cenário que foram usados nos exemplos anteriores.

No entanto, neste exemplo, a política da empresa é que a jornada de trabalho comece e termine às 7h.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `in_day`

Tabela de resultados

data	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

Como o argumento `start_day` de 7/24, que é 7h, é usado na função `inday()`, essa determina se cada data de transação ocorreu em 4 de janeiro, a partir das 7h, e em 5 de janeiro, antes das 7h.

Isso pode ser verificado na tabela de saída, onde as transações que ocorrem após 7h em 4 de janeiro retornam um resultado booleano de TRUE, enquanto as transações que ocorrem após 7h em 5 de janeiro retornam um resultado booleano de FALSE.

Exemplo 4 – Objeto de gráfico

Script de carregamento e expressão de gráfico

Visão geral

O script de carregamento usa o mesmo conjunto de dados e cenário que foram usados nos exemplos anteriores.

No entanto, neste exemplo, o conjunto de dados permanece inalterado e é carregado no aplicativo. Você calculará para determinar se uma transação ocorrerá em 5 de janeiro, criando uma medida em um objeto de gráfico.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão:

- date

Para calcular se uma transação ocorre em 5 de janeiro, crie a seguinte medida:

```
=inday(date,'01/05/2022 12:00:00 AM',0)
```

Tabela de resultados

data	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

Exemplo 5: Cenário

Script de carregamento e resultados

Visão geral

Neste exemplo, foi identificado que, devido a um erro de equipamento, os produtos fabricados em 5 de janeiro estavam com defeito. O usuário final gostaria de um objeto de gráfico que exiba, por data, o status de quais produtos fabricados estavam com defeito (Defective) ou sem falhas (Faultless), bem como o custo dos produtos fabricados em 5 de janeiro.

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados é carregado em uma tabela chamada "Products".
- A tabela contém os seguintes campos:
    - product ID
    - manufacture time
    - cost price

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão:

```
=dayname(manufacture_date)
```

Crie as seguintes medidas:

- `=if(only(InDay(manufacture_date,makedate(2022,01,05),0)),'Defective','Faultless')`
- `=sum(cost_price)`

Defina o **Formato numérico** da medida como **Dinheiro**.

Em **Aparência**, desative **Totais**.

Tabela de resultados

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

A função `inday()` retorna um valor booleano ao avaliar as datas de fabricação de cada um dos produtos. Para qualquer produto fabricado em 5 de janeiro, a função `inday()` retorna um valor booleano TRUE e marca os produtos como "Defective". Para qualquer produto que retorne um valor FALSE e, portanto, não tenha sido fabricado naquele dia, ele marca os produtos como "Faultless".

indaytotime

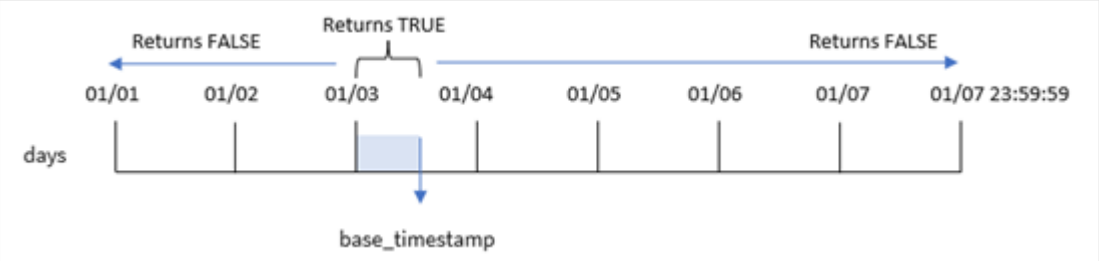
Esta função retorna True se **timestamp** estiver na parte do dia que contém a **base_timestamp** até e inclusive o exato milissegundo da **base_timestamp**.

Sintaxe:

```
InDayToTime (timestamp, base_timestamp, period_no[, day_start])
```

A função `indaytotime()` retorna um resultado booleano dependendo de quando um valor de carimbo de data/hora ocorre durante o segmento do dia. O limite inicial desse segmento é o início do dia, que é definido como meia-noite por padrão. O início do dia pode ser modificado pelo argumento `day_start` da função `indaytotime()`. O limite final do segmento do dia é determinado por um argumento `base_timestamp` da função.

Diagrama da função `indaytotime`.



Quando usar

A função `indaytotime()` retorna um resultado booleano. Normalmente, esse tipo de função será usado como uma condição em um `if expression`. A função `indaytotime()` retorna uma agregação ou cálculo dependendo de um carimbo de data/hora ter ocorrido no segmento do dia até e incluindo a hora do carimbo de data/hora base.

Por exemplo, a função `indaytotime()` pode ser usada para mostrar a soma das vendas de ingressos para shows que ocorreram até hoje.

Tipo de dados de retorno: Booleano

No Qlik Sense, o valor booleano “true” é representado por -1, e o valor falso é representado por 0.

Argumentos

Argumento	Descrição
`timestamp`	A data e hora que você deseja comparar com `base_timestamp`.
`base_timestamp`	Data e hora que são usadas para avaliar o carimbo de data/hora.
`period_no`	O dia pode ser deslocado por `period_no`. `period_no` é um inteiro, em que o valor 0 indica o dia que contém `base_timestamp`. Valores negativos em `period_no` indicam dias precedentes, e valores positivos indicam dias sucessivos.
`day_start`	(opcional) Se quiser trabalhar com dias que não começam à meia-noite, indique um deslocamento como uma fração de um dia em `day_start`. Por exemplo, use 0,125 para representar 3h

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplos de funções

Exemplo	Resultado
`indaytotime` ('01/12/2006 12:23:00 PM', '01/12/2006 11:59:00 PM', 0)	Retorna True

Exemplo	Resultado
indaytotime ('01/12/2006 12:23:00 PM', '01/12/2006 12:00:00 AM', 0)	Retorna False
indaytotime '01/11/2006 12:23:00 PM', '01/12/2006 11:59:00 PM', -1)	Retorna True

Exemplo 1: sem argumentos adicionais

Script de carregamento e resultados

Visão geral

Abra o Editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações para o período entre 4 e 5 de janeiro é carregado em uma tabela chamada "Transactions".
- Um campo de data que é fornecido na variável do sistema `TimeStamp`, no formato `(M/D/YYYY h:mm:ss [.fff] TT)`.
- Um carregamento anterior que contém a função `indaytotime()` definida como o campo `'in_day_to_time'`, que determina se cada uma das transações ocorrerá antes das 9h.

Script de carregamento

```
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
];
```

Resultados

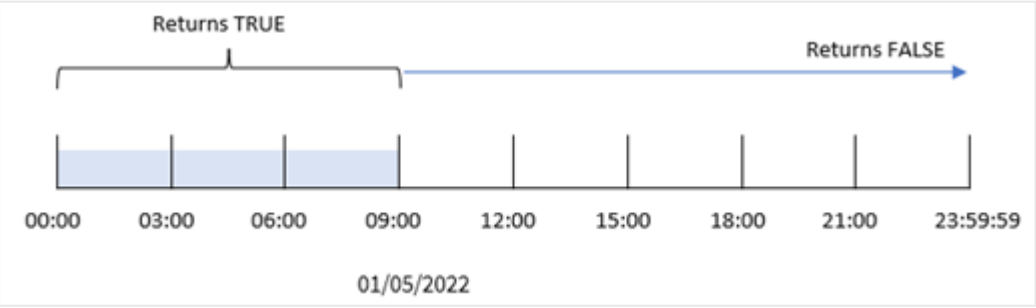
Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `in_day_to_time`

Tabela de resultados

data	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

Exemplo 1 – Diagrama da função `indaytotime` com limite de 9h.



O campo `in_day_to_time field` é criado na instrução de carregamento anterior usando a função `indaytotime()` e transmitindo o campo de data, um carimbo de data/hora codificado para 9h de 5 de janeiro e um deslocamento de 0 como argumentos da função. Todas as transações que ocorrerem entre a meia-noite e as 9h do dia 5 de janeiro retornarão TRUE.

Exemplo 2 – period_no

Script de carregamento e resultados

Visão geral

O script de carregamento usa o mesmo conjunto de dados e cenário que foram usados no primeiro exemplo.

No entanto, neste exemplo, você calculará se a data da transação ocorreu um dia antes das 9h do dia 5 de janeiro.

Script de carregamento

```
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
];
```

Resultados

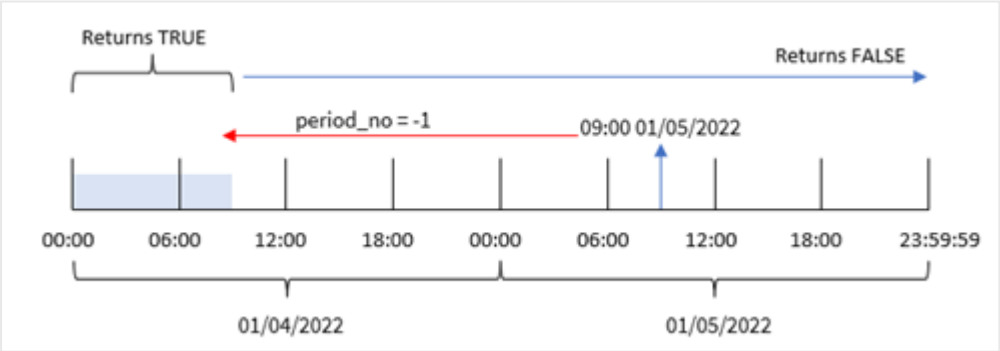
Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- date
- in_day_to_time

Tabela de resultados

data	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

Exemplo 2 – Diagrama de função `indaytotime` com transações de 4 de janeiro.



Neste exemplo, como um deslocamento de -1 foi usado como argumento de deslocamento na função `indaytotime()`, ela determina se cada data de transação ocorreu antes das 9h de 4 de janeiro. Isso pode ser verificado na tabela de saída em que uma transação retorna um resultado booleano TRUE.

Exemplo 3 – day_start

Script de carregamento e resultados

Visão geral

São usados o mesmo conjunto de dados e cenário do primeiro exemplo.

No entanto, neste exemplo, a política da empresa é que a jornada de trabalho comece e termine às 8h.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

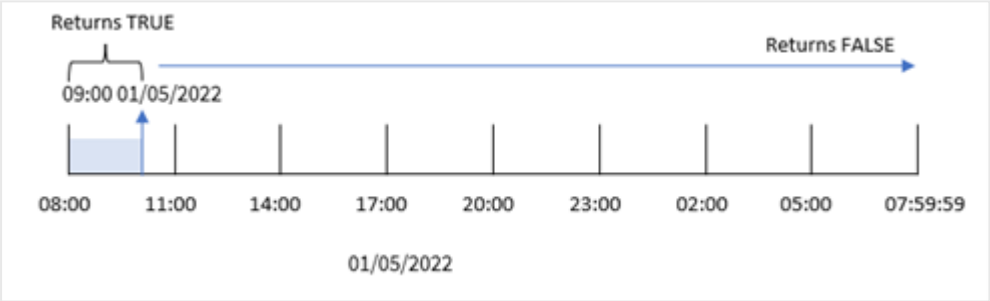
- date
- in_day_to_time

Tabela de resultados

data	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

data	in_day_to_time
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

Exemplo 3 – Diagrama da função `indaytotime` com transações das 8h às 9h.



Como o argumento `start_day` de 8/24, que equivale a 8h, é usado na função `indaytotime()`, cada dia começa e termina às 8h. Portanto, a função `indaytotime()` retornará um resultado booleano TRUE para qualquer transação que tenha ocorrido entre 8h e 9h em 5 de janeiro.

Exemplo 4 – Objeto de gráfico

Script de carregamento e expressão de gráfico

Visão geral

São usados o mesmo conjunto de dados e cenário do primeiro exemplo.

No entanto, neste exemplo, o conjunto de dados permanece inalterado e é carregado no aplicativo. Você calculará para determinar se uma transação ocorrerá em 5 de janeiro antes das 9h, criando uma medida em um objeto de gráfico.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão:

`date`.

Para determinar se uma transação ocorre em 5 de janeiro antes das 9h, crie a seguinte medida:

```
=indaytotime(date,'01/05/2022 9:00:00 AM',0)
```

Tabela de resultados

data	**=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

A medida `in_day_to_time` é criada no objeto de gráfico usando a função `indaytotime()` e transmitindo o campo de data, um carimbo de data/hora codificado para 9h em 5 de janeiro e um deslocamento de `0` como argumentos da função. Todas as transações que ocorrerem entre a meia-noite e as 9h do dia 5 de janeiro retornarão TRUE. Isso é validado na tabela de resultados.

Exemplo 5: Cenário

Script de carregamento e resultados

Visão geral

Neste exemplo, um conjunto de dados contendo vendas de ingressos para um cinema local é carregado em uma tabela chamada Ticket_Sales. Hoje é 3 de maio de 2022 e são 11:00 da manhã.

O usuário gostaria que um objeto de gráfico de KPI mostrasse a receita obtida de todos os programas que ocorreram até agora.

Script de carregamento

```
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
];
```

Resultados

Faça o seguinte:

1. Crie um objeto de KPI.
2. Crie uma medida que mostre a soma de todas as vendas de ingressos para exibições que ocorreram hoje até agora usando a função `indaytotime()`:

```
=sum(if(indaytotime([show time],'05/03/2022 11:00:00 AM',0),[ticket price],0))
```

3. Crie um rótulo para o objeto de KPI, "Current Revenue".
4. Defina a **Formatação numérica** da medida como **Dinheiro**.

A soma total das vendas de ingressos até 11h em 3 de maio de 2022 é de $52,50.

A função `indaytotime ()` retorna um valor booleano ao comparar os horários de exibição de cada uma das vendas de ingressos com a hora atual ('05/03/2022 11:00:00 AM'). Para qualquer exibição em 3 de maio antes das 11h, a função `indaytotime ()` retorna um valor booleano de TRUE, e o preço do ingresso será incluído na soma total.

inlunarweek

Essa função determina se **timestamp** está dentro da semana lunar que contém **base_date**. As semanas lunares no Qlik Sense são definidas contando 1º de janeiro como o primeiro dia da semana. Além da última semana do ano, cada semana conterá exatamente sete dias.

Sintaxe:

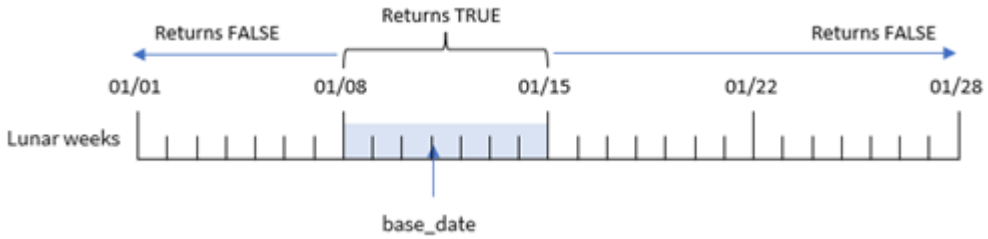
```
InLunarWeek (timestamp, base_date, period_no[, first_week_day])
```

Tipo de dados de retorno: Booleano

> *No Qlik Sense, o valor booleano "true" é representado por -1, e o valor falso é representado por 0.*

A função `inlunarweek()` determina em qual semana lunar a `base_date` cai. Em seguida, ela retorna um resultado booleano depois de determinar se cada valor de carimbo de data/hora ocorre durante a mesma semana lunar que `base_date`.

Diagrama da função `inlunarweek()`



Quando usar

A função `inlunarweek()` retorna um resultado booleano. Normalmente, esse tipo de função será usado como uma condição em uma expressão IF. Isso retornaria uma agregação ou cálculo dependendo se a data avaliada ocorreu durante a semana lunar em questão.

Por exemplo, a função `inlunarweek()` pode ser usada para identificar todos os equipamentos fabricados em uma determinada semana lunar.

Argumentos

Argumento	Descrição
timestamp	A data que você deseja comparar com **base_date**.
base_date	Data que é usada para avaliar a semana lunar.
period_no	A semana lunar pode ser deslocada por **period_no**. period_n é um inteiro, em que o valor 0 indica a semana lunar que contém **base_date**. Valores negativos em **period_no** indicam semanas lunares precedentes e valores positivos indicam semanas lunares sucessivas.
first_week_day	Um deslocamento que pode ser maior ou menor que zero. Isso muda o início do ano por um número determinado de dias e/ou frações de um dia.

Exemplos de funções

Exemplo	Resultado
`inlunarweek ('01/12/2013', '01/14/2013', 0)`	Retorna TRUE, já que o valor de `timestamp`, 01/12/2013, cai na semana de 01/08/2013 a 01/14/2013.
`inlunarweek ('01/12/2013', '01/07/2013', 0)`	Retorna FALSE, já que `base_date` 01/07/2013 está na semana lunar definida como 01/01/2013 a 01/07/2013.
`inlunarweek ('01/12/2013', '01/14/2013', -1)`	Retorna FALSE. Especificar um valor de period_no como -1 muda a semana para a semana anterior, 01/01/2013 para 01/07/2013.
`inlunarweek ('01/07/2013', 01/14/2013', -1)`	Retorna TRUE. Em comparação com o exemplo anterior, `timestamp` está na semana seguinte, depois de considerar a mudança para trás.
`inlunarweek ('01/11/2006', '01/08/2006', 0, 3)`	Retorna FALSE. Especificar um valor de 3 para `first_week_day` significa que o início do ano é calculado a partir de 01/04/2013. Portanto, o valor de `base_date` cai na primeira semana, e o valor de `timestamp` cai na semana de 01/11/2013 a 01/17/2013.

A função `inlunarweek()` é frequentemente usada em combinação com as seguintes funções:

Funções relacionadas

Função	Interação
lunarweekname (page 866)	Essa função é usada para determinar o número da semana lunar do ano em que ocorre uma data de entrada.

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplo 1: Nenhum argumento adicional

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados de transações para o mês de janeiro, que é carregado em uma tabela denominada `Transactions`.
- O campo de data foi fornecido no formato da variável de sistema `DateFormat` (MM/DD/AAAA).

Crie um campo, `in_lunar_week`, que determina se as transações ocorreram na mesma semana lunar de 10 de janeiro.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

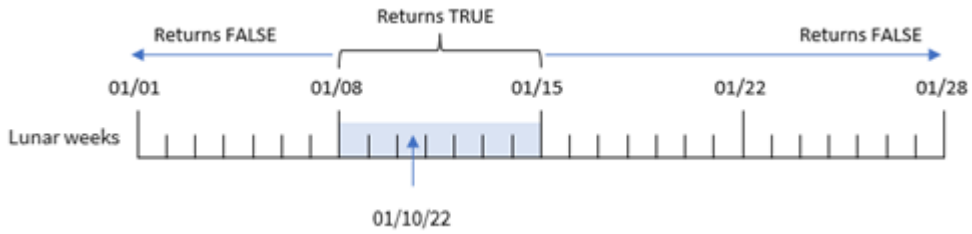
- `date`
- `in_lunar_week`

Tabela de resultados

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

date	in_lunar_week
1/19/2022	0
1/20/2022	0
1/21/2022	0
1/22/2022	0
1/23/2022	0

Função `inlunarweek()`, exemplo básico



O campo `in_lunar_week` é criado na instrução de carregamento anterior usando a função `inlunarweek()` e transmitindo o seguinte como argumentos da função:

- O campo `date`
- Uma data codificada para 10 de janeiro como `base_date`
- Um `period_no` de 0

Como as semanas lunares começam em 1º de janeiro, 10 de janeiro cairia na semana lunar que começa em 8 de janeiro e termina em 14 de janeiro. Portanto, qualquer transação que ocorra entre essas duas datas em janeiro retornaria um valor booleano de `TRUE`. Isso é validado na tabela de resultados.

Exemplo 2: period_no

Exemplos e resultados:

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- O mesmo conjunto de dados e cenário do primeiro exemplo.
- O campo de data foi fornecido no formato da variável de sistema `DateFormat` (MM/DD/AAAA).

No entanto, neste exemplo, a tarefa é criar um campo, `2_lunar_weeks_later`, que determina se as transações ocorreram ou não duas semanas lunares após 10 de janeiro.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

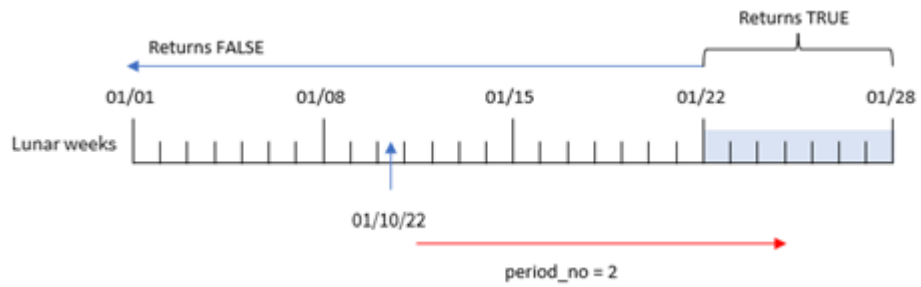
- `date`
- `2_lunar_weeks_later`

Tabela de resultados

date	2_lunar_weeks_later
1/5/2022	0
1/6/2022	0
1/7/2022	0
1/8/2022	0

date	2_lunar_weeks_later
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

função `inlunarweek()`, exemplo de period_no



Nesse caso, como uma `period_no` de 2 foi usada como argumento offset na função `inlunarweek()`, a função define a semana que começa em 22 de janeiro como a semana lunar para validar transações. Portanto, qualquer transação que ocorra entre 22 e 28 de janeiro retornará um resultado booleano de TRUE.

Exemplo 3: first_week_day

Script de carregamento e resultados

Visão geral

Abra o editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento usa o mesmo conjunto de dados e cenário do primeiro exemplo. No entanto, no exemplo, definimos que as semanas lunares começam em 6 de janeiro.

- O mesmo conjunto de dados e cenário do primeiro exemplo.
- A variável de sistema `DateFormat` padrão MM/DD/AAAA é usada.
- Um argumento `first_week_day` de 5. Isso define que as semanas lunares comecem em 5 de janeiro.

Script de carregamento

```
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
];
```

Resultados

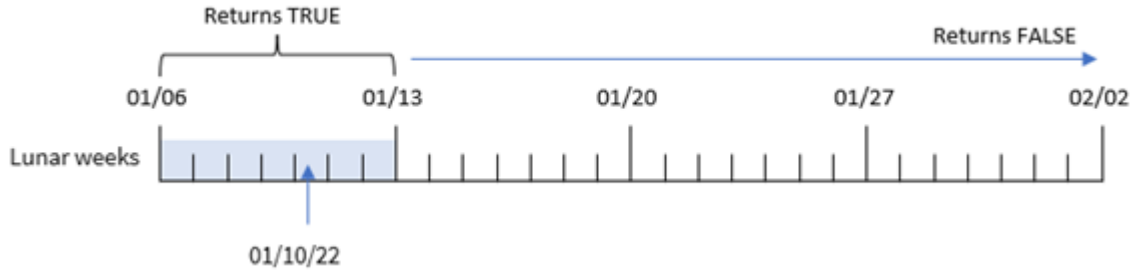
Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `in_lunar_week`

Tabela de resultados

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
1/19/2022	0
1/20/2022	0
1/21/2022	0
1/22/2022	0
1/23/2022	0

função `inlunarweek()`, exemplo de first_week_day



Nesse caso, como o argumento `first_week_date` de 5 é usado na função `inlunarweek()`, ele desloca o início do calendário da semana lunar para 6 de janeiro. Portanto, 10 de janeiro cai na semana lunar que começa em 6 de janeiro e termina em 12 de janeiro. Qualquer transação que esteja entre essas duas datas retornará um valor booleano de TRUE.

Exemplo 4: Objeto de gráfico

Script de carregamento e expressão de gráfico:

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- O mesmo conjunto de dados e cenário do primeiro exemplo.
- O campo de data foi fornecido no formato da variável de sistema `DateFormat` (MM/DD/AAAA).

No entanto, neste exemplo, o conjunto de dados inalterado é carregado no aplicativo. O cálculo que determina se as transações ocorreram na mesma semana lunar de 10 de janeiro é criado como uma medida em um objeto de gráfico do aplicativo.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão: `date`.

Para calcular se uma transação ocorre na semana lunar que contém 10 de janeiro, crie a seguinte medida:

```
= inlunarweek(date,'01/10/2022', 0)
```

Tabela de resultados

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
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

Exemplo 5: Cenário

Script de carregamento e expressão de gráfico:

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados que é carregado em uma tabela denominada `Products`.
- Informações que consistem em ID do produto, data de fabricação e preço de custo.

Foi identificado que, devido a um erro de equipamento, os produtos que foram fabricados na semana lunar que inclui 12 de janeiro estavam com defeito. O usuário final deseja um objeto de gráfico que mostre, por nome da semana lunar, o status que indica se os produtos fabricados estavam "com defeito" ou "sem defeito", bem como o custo dos produtos fabricados naquele mês.

Script de carregamento

```
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
];
```

Resultados

Faça o seguinte:

1. Carregue os dados e abra uma pasta. Crie uma nova tabela.
2. Crie uma dimensão para mostrar os nomes dos meses:
   `=lunarweekname(manufacture_date)`
3. Crie uma medida para identificar quais produtos estão com defeito e quais estão sem defeito usando a função `inlunarweek()`:
   `=if(only(inlunarweek(manufacture_date,makedate(2022,01,12),0)), 'Defective','Faultless')`
4. Crie uma medida para somar o `cost_price` dos produtos:
   `=sum(cost_price)`
5. Defina o **Formato numérico** da medida como **Dinheiro**.
6. Em **Aparência**, desative **Totais**.

Tabela de resultados

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

A função `inlunarweek()` retorna um valor booleano ao avaliar as datas de fabricação de cada um dos produtos. Para qualquer produto fabricado na semana lunar que contém 10 de janeiro, a função `inlunarweek()` retorna um valor booleano de TRUE e marca os produtos como "com defeito". Para qualquer produto que retorne um valor de FALSE e, portanto, não tenha sido fabricado nessa semana, ela marca os produtos como "sem defeito".

inlunarweektodate

Esta função descobre se **timestamp** está dentro da parte da semana lunar até e inclusive o último milissegundo da **base_date**. As semanas lunares no Qlik Sense são definidas contando 1º de janeiro como o primeiro dia da semana e, além da última semana do ano, conterão exatamente sete dias.

Sintaxe:

```
InLunarWeekToDate (timestamp, base_date, period_no [, first_week_day])
```

Tipo de dados de retorno: Booleano

No Qlik Sense, o valor booleano "true" é representado por -1, e o valor falso é representado por 0.

Exemplo de diagrama da função `inlunarweektodate()`

A função `inlunarweektodate()` atua como o ponto final da semana lunar. Por outro lado, a função `inlunarweek()` determina em qual semana lunar a `base_date` cai. Por exemplo, se `base_date` fosse 5 de janeiro, qualquer carimbo de data/hora entre 1º de janeiro e 5 de janeiro retornaria um resultado booleano de TRUE, enquanto as datas de 6 e 7 de janeiro e posteriores retornariam um resultado booleano de FALSE.

Argumentos

Argumento	Descrição
timestamp	A data que você deseja comparar com **base_date**.
base_date	Data que é usada para avaliar a semana lunar.
period_no	A semana lunar pode ser deslocada por **period_no**. period_n é um inteiro, em que o valor 0 indica a semana lunar que contém **base_date**. Valores negativos em **period_no** indicam semanas lunares precedentes e valores positivos indicam semanas lunares sucessivas.
first_week_day	Um deslocamento que pode ser maior ou menor que zero. Isso muda o início do ano por um número determinado de dias e/ou frações de um dia.

Quando usar

A função `inlunarweektodate()` retorna um resultado booleano. Normalmente, esse tipo de função será usado como uma condição em uma expressão IF. A função `inlunarweektodate()` seria usada quando o usuário quisesse que o cálculo retornasse uma agregação ou um cálculo, dependendo se a data avaliada ocorreu durante um determinado segmento da semana em questão.

Por exemplo, a função `inlunarweektodate()` pode ser usada para identificar todos os equipamentos fabricados em uma determinada semana até e incluindo uma data específica.

Exemplos de funções

Exemplo	Resultado
`inlunarweektodate ('01/12/2013', '01/13/2013', 0)`	Retorna `TRUE`, já que o valor de `timestamp`, 01/12/2013, cai na parte da semana de 01/08/2013 a 01/13/2013.
`inlunarweektodate ('01/12/2013', '01/11/2013', 0)`	Retorna `FALSE`, já que o valor de `timestamp` é posterior ao valor de `base_date`, mesmo que as duas datas estejam na mesma semana lunar anterior a 01/12/2012.
`inlunarweektodate ('01/12/2006', '01/05/2006', 1)`	Retorna `TRUE`. Especificar um valor de 1 para `period_no` desloca a `base_date` para frente uma semana, assim, o valor de `timestamp` cai na parte da semana lunar.

A função `inlunarweektodate()` é frequentemente usada em combinação com as seguintes funções:

Funções relacionadas

Função	Interação
lunarweekname (page 866)	Essa função é usada para determinar o número da semana lunar do ano em que ocorre uma data de entrada.

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações

regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplo 1: Nenhum argumento adicional

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações para o mês de janeiro, que é carregado em uma tabela denominada `Transactions`. A `variável` de sistema DateFormat padrão MM/DD/AAAA é usada.
- Crie um campo, `in_lunar_week_to_date`, que determina quais transações ocorreram na semana lunar até a data de 10 de janeiro.

Script de carregamento

```
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
];
```

Resultados

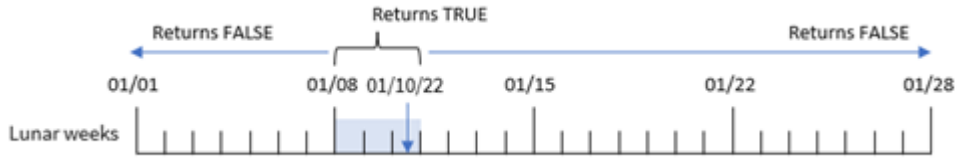
Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `in_lunar_week_to_date`

Tabela de resultados

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

Função `inlunarweektodate()`, sem argumentos adicionais



O campo `in_lunar_week_to_date` é criado na instrução de carregamento anterior usando a função `inlunarweektodate()` e transmitindo o campo `date`, uma data codificada para 10 de janeiro como `base_date` e um deslocamento de 0 como argumentos da função.

Como as semanas lunares começam em 1º de janeiro, 10 de janeiro cairia na semana lunar que começa em 8 de janeiro, e, como estamos usando a função `inlunarweektodate()`, essa semana lunar terminaria no dia 10. Portanto, qualquer transação que ocorra entre essas duas datas em janeiro retornaria um valor booleano de `TRUE`. Isso é validado na tabela de resultados.

Exemplo 2: period_no

Script de carregamento e resultados

Visão geral

Abra o editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém o mesmo conjunto de dados e cenário do primeiro exemplo. No entanto, neste exemplo, a tarefa é criar um campo, `2_lunar_weeks_later`, que determina se as transações ocorreram ou não duas semanas após a semana lunar até a data de 1º de janeiro.

Script de carregamento

```
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
];
```

Resultados

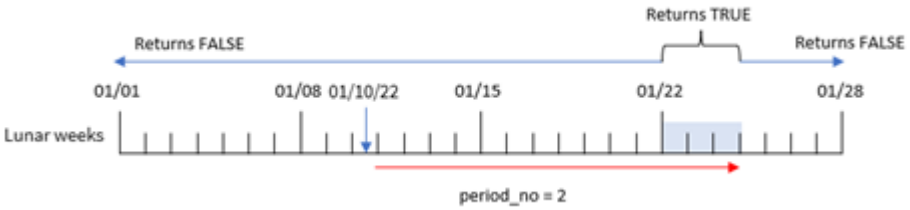
Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `2_lunar_weeks_later`

Tabela de resultados

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

função `inlunarweektodate()`, exemplo de period_no



Nesse caso, a função `inlunarweektodate()` determina que a semana lunar até 10 de janeiro equivale a três dias (8, 9, 10 de janeiro). Como um `period_no` de 2 foi usado como argumento offset, essa semana lunar é alterada em 14 dias. Portanto, isso define que a semana lunar de três dias incluirá 22, 23 e 24 de janeiro. Qualquer transação que ocorra entre 22 de janeiro e 24 de janeiro retornará um resultado booleano de TRUE.

Exemplo 3: first_week_day

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- O mesmo conjunto de dados e cenário do primeiro exemplo.
- A variável de sistema `DateFormat` padrão MM/DD/AAAA é usada.
- Um argumento `first_week_date` de 3. Isso define que as semanas lunares comecem em 3 de janeiro.

Script de carregamento

```
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
];
```

Resultados

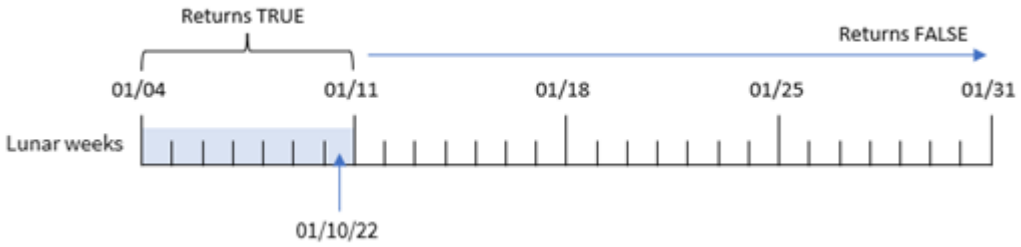
Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `in_lunar_week_to_date`

Tabela de resultados

date	in_lunar_week_to_date
1/1/2022	0
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

função `inlunarweektodate()`, exemplo de first_week_day



Nesse caso, como o argumento `the first_week_date` de 3 é usado na função `inlunarweek()`, a primeira semana lunar será de 3 de janeiro a 10 de janeiro. Como 10 de janeiro também é `base_date`, qualquer transação que ocorra entre essas duas datas retornará um valor booleano de `TRUE`.

Exemplo 4: Exemplo de objeto de gráfico

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém o mesmo conjunto de dados e cenário do primeiro exemplo.

No entanto, neste exemplo, o conjunto de dados inalterado é carregado no aplicativo. O cálculo que determina se as transações ocorreram na semana lunar até 10 de janeiro é criado como uma medida em um objeto de gráfico do aplicativo.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão: `date`.

Crie a seguinte medida:

```
=inlunarweektodate(date,'01/10/2022', 0)
```

Tabela de resultados

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

Função `inlunarweektodate()`, exemplo de objeto de gráfico

A medida `in_lunar_week_to_date` é criada no objeto de gráfico usando a função `inlunarweektodate()` e transmitindo o campo de data, uma data codificada para 10 de janeiro como `base_date` e um deslocamento de 0 como os argumentos da função.

Como as semanas lunares começam em 1º de janeiro, 10 de janeiro cairia na semana lunar que começa em 8 de janeiro. Além disso, como estamos usando a função `inlunarweektodate()`, essa semana lunar terminaria no dia 10. Portanto, qualquer transação que ocorra entre essas duas datas em janeiro retornaria um valor booleano de `TRUE`. Isso é validado na tabela de resultados.

Exemplo 5: Cenário

Script de carregamento e expressões de gráfico

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados que é carregado em uma tabela denominada `Products`.
- Informações que consistem em ID do produto, data de fabricação e preço de custo.

Foi identificado que, devido a um erro de equipamento, os produtos fabricados na semana lunar de 12 de janeiro estavam com defeito. O problema foi resolvido em 13 de janeiro. O usuário final deseja um objeto de gráfico que mostre, por semana, o status que indica se os produtos fabricados estão "com defeito" ou "sem defeito", bem como o custo dos produtos fabricados naquela semana.

Script de carregamento

```
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
];
```

Resultados

Faça o seguinte:

1. Carregue os dados e abra uma pasta. Crie uma nova tabela.
2. Crie uma dimensão para mostrar os nomes das semanas:
   `=weekname(manufacture_date)`
3. Em seguida, crie uma dimensão que use a função `inlunarweektodate()` para identificar quais produtos estão com defeito e quais estão sem falhas:
   `=if(inlunarweektodate(manufacture_date,makedate(2022,01,12),0),'Defective','Faultless')`
4. Crie uma medida para somar o `cost_price` dos produtos:
   `=sum(cost_price)`
5. Defina o **Formato numérico** da medida como **Dinheiro**.

Tabela de resultados

=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

A função `inlunarweektodate()` retorna um valor booleano ao avaliar as datas de fabricação de cada um dos produtos. Para os casos que retornam um valor booleano de `TRUE`, ela marca os produtos como `'Defective'`. Para qualquer produto que retorne um valor de `FALSE` e, portanto, não tenha sido fabricado na semana lunar até 12 de janeiro, ela marca os produtos como `'Faultless'`.

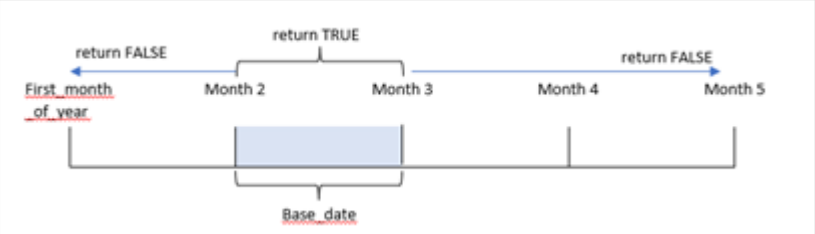
inmonth

Esta função retornará True se **timestamp** estiver dentro do mês que contém **base_date**.

Sintaxe:

```
InMonth (timestamp, base_date, period_no)
```

Diagrama da função `indaytotime`.



Em outras palavras, a função `inmonth()` determina se um conjunto de datas cai neste mês e retorna um valor booleano com base em uma `base_date` que identifica o mês.

Quando usar

A função `inmonth()` retorna um resultado booleano. Normalmente, esse tipo de função será usado como uma condição em um `if expression`. Isso retorna uma agregação ou cálculo dependendo de se uma data ocorreu no mês, incluindo a data em questão.

Por exemplo, a função `inmonth()` pode ser usada para identificar todos os equipamentos fabricados em um mês específico.

Tipo de dados de retorno: Booleano

No Qlik Sense, o valor booleano “true” é representado por -1, e o valor falso é representado por 0.

Argumentos

Argumento	Descrição
carimbo de data/hora	A data que você deseja comparar com `base_date`.
base_date	Data que é usada para avaliar o mês. É importante notar que `base_date` pode ser qualquer dia dentro de um mês.
period_no	O mês pode ser deslocado por `period_no`. `period_no` é um inteiro, em que o valor 0 indica o mês que contém `base_date`. Valores negativos em `period_no` indicam meses precedentes, e valores positivos indicam meses sucessivos.

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplos de funções

Exemplo	Resultado
`inmonth ('25/01/2013', '01/01/2013', 0)`	Retorna True
`inmonth('25/01/2013', '23/04/2013', 0)`	Retorna False
`inmonth ('25/01/2013', '01/01/2013', -1)`	Retorna False
`inmonth ('25/12/2012', '17/01/2013', -1)`	Retorna True

Exemplo 1: Sem argumentos adicionais

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações para o primeiro semestre de 2022.
- Um carregamento anterior com uma variável adicional "in_month", que determina se as transações ocorreram em abril.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- date
- in_month

Exemplos de funções

date	in_month
1/10/2022	0
1/14/2022	0
1/20/2022	0
1/22/2022	0

date	in_month
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

O campo "in_month" é criado na instrução de carregamento anterior usando a função `inmonth()` e transmitindo o campo de data, uma data codificada de 1º de abril, como nossa `base_date` e um `period_no` de `0` como argumentos da função.

A `base_date` identifica o mês que retornará um resultado booleano de TRUE. Portanto, todas as transações que ocorreram em abril retornam TRUE, que é validado na tabela de resultados.

Exemplo 2: period_no

Script de carregamento e resultados

Visão geral

São usados o mesmo conjunto de dados e cenário do primeiro exemplo.

No entanto, neste exemplo, você criará um campo, "`2_months_prior`", que determina se as transações ocorreram dois meses antes de abril.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `2_months_prior`

Exemplos de funções

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

Usar -2 como argumento `period_no` na função `inmonth()` muda o mês definido pelo argumento `base_date` dois meses antes. Neste exemplo, ele altera o mês definido de abril para fevereiro.

Portanto, qualquer transação que ocorra em fevereiro retornará um resultado booleano de TRUE.

Exemplo 3: Objeto de gráfico

Script de carregamento e expressão de gráfico

Visão geral

São usados o mesmo conjunto de dados e cenário dos exemplos anteriores.

No entanto, neste exemplo, o conjunto de dados permanece inalterado e é carregado no aplicativo. O cálculo que determina se as transações ocorreram em abril é criado como uma medida em um objeto de gráfico do aplicativo.

Script de carregamento

```
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
];
```

Objeto de gráfico

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão:

```
date
```

Para calcular se uma transação ocorre em abril, crie a seguinte medida:

```
=inmonth(date,'04/01/2022', 0)
```

Resultados

Exemplos de funções

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

Exemplo 4: Cenário

Script de carregamento e resultados

Visão geral

Neste exemplo, um conjunto de dados é carregado em uma tabela chamada "`Products`". A tabela contém os seguintes campos:

- ID do produto
- Data de fabricação
- Preço de custo

Devido a erros de equipamentos, os produtos fabricados no mês de julho de 2022 estavam com defeito. O problema foi resolvido em 27 de julho de 2022.

O usuário final deseja um gráfico que mostre, por mês, o status dos produtos que foram fabricados como "com defeito" (booleano TRUE) ou "sem defeito" (booleano FALSE) e também o custo dos produtos fabricados naquele mês.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão:

```
=monthname(manufacture_date)
```

Crie as seguintes medidas

- `=sum(cost_price)`
- `=if(only(inmonth(manufacture_date,makedate(2022,07,01),0)),'Defective','Faultless')`

1. Defina o **Formato numérico** da medida como **Dinheiro**.
2. Em **Aparência**, desative **Totais**.

Tabela de resultados

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
Mar 2022	Faultless	$53.80

monthname (manufacture_date)	=if(only(inmonth(manufacture_date,makedate (2022,07,01),0)),'Defective','Faultless')	sum(cost_ price)
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

A função `inmonth()` retorna um valor booleano ao avaliar as datas de fabricação de cada um dos produtos. Para qualquer produto fabricado em julho de 2022, a função `inmonth()` retorna um valor booleano True e marca os produtos como "Defective". Para qualquer produto que retorne um valor de False e, portanto, não tenha sido fabricado em julho, ele marca os produtos como "Faultless".

inmonths

Essa função descobre se um carimbo de data/hora está dentro do mesmo mês, bimestre, trimestre, quadrimestre ou semestre como data base. Também é possível descobrir se uma data/hora está dentro de um período de tempo anterior ou seguinte.

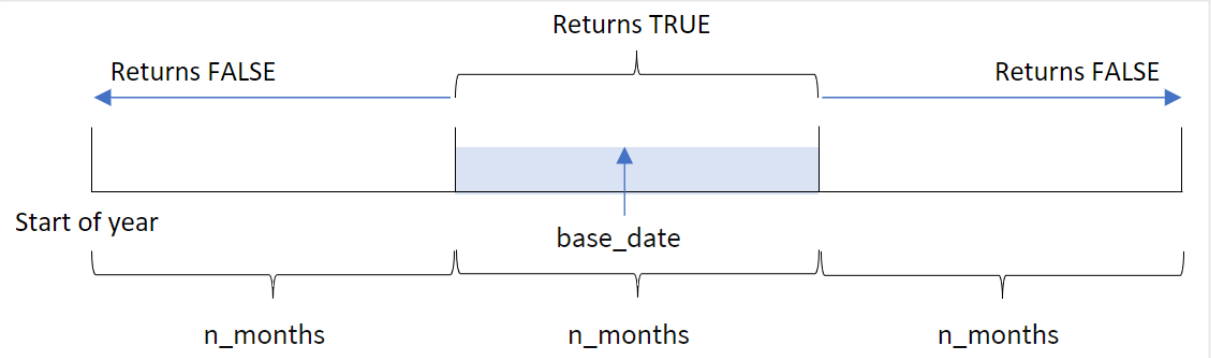
Sintaxe:

```
InMonths(n_months, timestamp, base_date, period_no [, first_month_of_year])
```

Tipo de dados de retorno: Booleano

No Qlik Sense, o valor booleano “true” é representado por -1, e o valor falso é representado por 0.

Diagrama da função `inmonths()`



A função `inmonths()` divide o ano em segmentos com base no argumento `n_months` fornecido. Em seguida, ela determina se cada carimbo de data/hora avaliado se enquadra no mesmo segmento do argumento `base_date`. No entanto, se um argumento `period_no` for fornecido, a função determinará se os carimbos de data/hora se enquadram em um período anterior ou posterior de `base_date`.

Os seguintes segmentos do ano estão disponíveis na função como argumentos `n_month`.

Argumentos `n_month`

Período	Número de meses
mês	1
bimestre	2
trimestre	3
quadrimestre	4
semestre	6

Quando usar

A função `inmonths()` retorna um resultado booleano. Normalmente, esse tipo de função será usado como uma condição em um `if expression`. Usando a função `inmonths()`, você pode selecionar o período que deseja avaliar. Por exemplo, permitir que o usuário identifique produtos fabricados no mês, trimestre ou semestre de um determinado período.

Tipo de dados de retorno: Booleano

No Qlik Sense, o valor booleano “true” é representado por -1, e o valor falso é representado por 0.

Argumentos

Argumento	Descrição
n_months	O número de meses que define o período. Um inteiro ou uma expressão que se resolve como um inteiro que deve ser um dos seguintes: 1 (equivalente à função inmonth()), 2 (bimestre), 3 (equivalente à função inquarter()), 4 (quadrimestre) ou 6 (semestre).
timestamp	A data que você deseja comparar com **base_date**.
base_date	Data que é usada para avaliar o período.
period_no	O período pode ser deslocado por **period_no**, um inteiro ou expressão que resolve um inteiro, no qual o valor 0 indica o período que contém **base_date**. Valores negativos em **period_no** indicam períodos precedentes e valores positivos indicam períodos sucessivos.
first_month_of_year	Se desejar trabalhar com anos (fiscais) que não comecem em janeiro, indique um valor entre 2 e 12 em **first_month_of_year**.

Você pode usar os seguintes valores para definir o primeiro mês do ano no argumento `first_month_of_year`:

Valores first_month_of_year

Month	Valor
Fevereiro	2
Março	3
Abril	4
Maio	5
Junho	6
Julho	7
Agosto	8
Setembro	9
Outubro	10
Novembro	11
Dezembro	12

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplos de funções

Exemplo	Resultado
`inmonths(4, '01/25/2013', '04/25/2013', 0)`	Retorna TRUE. Porque o valor de timestamp, 25/01/2013, está dentro do período de quatro meses de 01/01/2013 a 30/04/2013, no qual está o valor de `base_date`, 25/04/2013.
`inmonths(4, '05/25/2013', '04/25/2013', 0)`	Retorna FALSE. Porque 25/05/2013 está fora do mesmo período do exemplo anterior.

Exemplo	Resultado
`inmonths(4, '11/25/2012', '02/01/2013', -1 )`	Retorna TRUE. Porque o valor de `period_no`, `-1`, retrocede o período de pesquisa em um período de quatro meses (o valor de n meses), o que torna o período de pesquisa de 01/09/2012 a 31/12/2012.
`inmonths(4, '05/25/2006', '03/01/2006', 0, 3)`	Retorna TRUE. Como o valor de `first_month_of_year` está definido como `3`, o que torna o período de pesquisa de 01/03/2006 a 30/07/2006 em vez de 01/01/2006 a 30/04/2006.

Exemplo 1: Nenhum argumento adicional

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações para 2022 é carregado em uma tabela denominada "`Transactions`".
- Um carregamento anterior com uma variável adicional, "`in_months`", que determina quais transações ocorreram no mesmo trimestre de 15 de maio de 2022.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

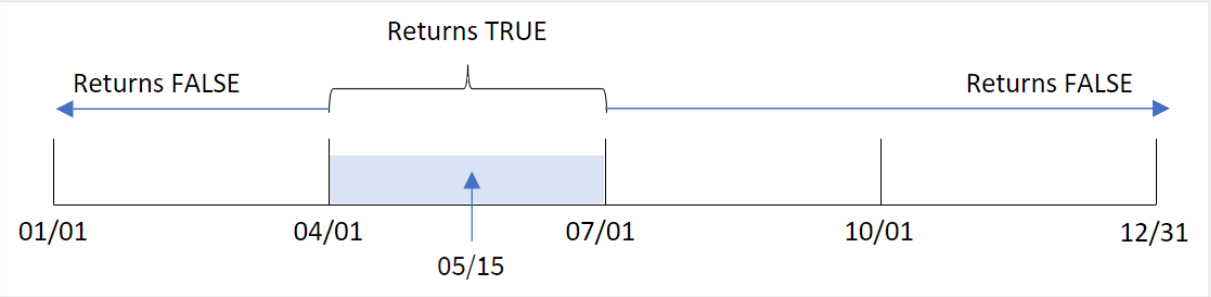
- `date`
- `in_months`

Tabela de resultados

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
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

O campo "in_months" é criado na instrução de carregamento anterior usando a função `inmonths()`. O primeiro argumento fornecido é `3`, que divide o ano em segmentos trimestrais. O segundo argumento identifica qual campo está sendo avaliado, o campo de data neste exemplo. O terceiro argumento é uma data codificada para 15 de maio, que é `base_date`, e `period_no` de `0` é o argumento final.

Diagrama da função `inmonths()` com segmentos trimestrais



Maio cai no segundo trimestre do ano. Portanto, qualquer transação que ocorra entre 1º de abril e 30 de junho retornará um resultado booleano de TRUE. Isso é validado na tabela de resultados.

Exemplo 2: period_no

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações para 2022, que é carregado em uma tabela denominada "`Transactions`".
- Um carregamento anterior com uma variável adicional, “`previous_quarter`”, que determina se as transações ocorreram no trimestre anterior a 15 de maio de 2022.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `previous_quarter`

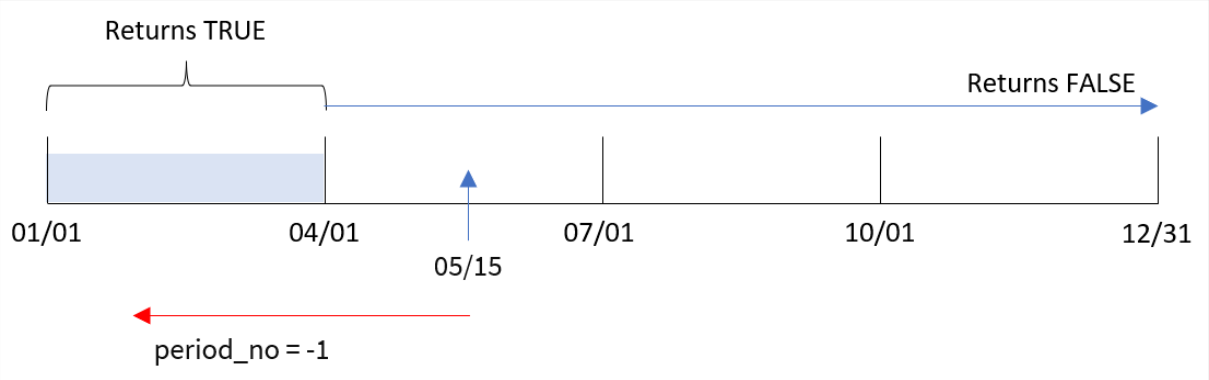
Tabela de resultados

date	previous quarter
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
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0

date	previous quarter
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

A função avalia se as transações ocorreram no primeiro trimestre do ano usando `-1` como argumento `period_no` na função `inmonths()`. 15 de maio é `base_date` e cai no segundo trimestre do ano (abril-junho).

Diagrama da função `inmonths()` com segmentos trimestrais e `period_no` definido como `-1`



Portanto, qualquer transação que ocorra entre janeiro e março retornará um resultado booleano de TRUE.

Exemplo 3: first_month_of_year

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações para 2022, que é carregado em uma tabela denominada "`Transactions`".
- Um carregamento anterior com uma variável adicional, "`in_months`", que determina quais transações ocorreram no mesmo trimestre de 15 de maio de 2022.

Neste exemplo, a política organizacional é que março seja o primeiro mês do exercício financeiro.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `in_months`

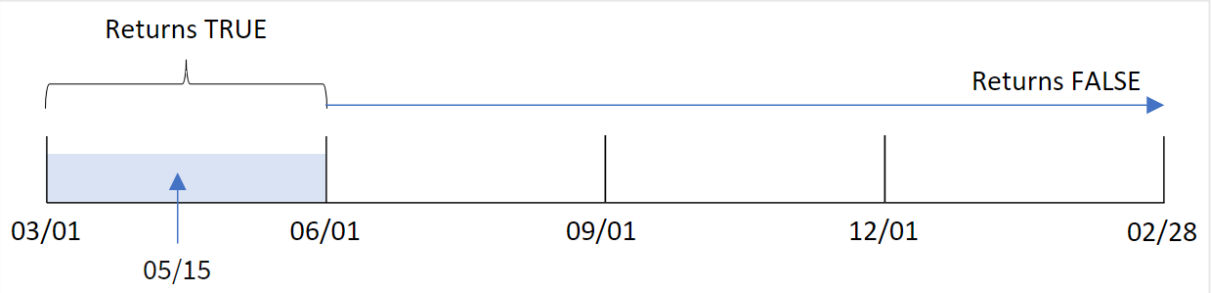
Tabela de resultados

date	in_months
2/19/2022	0
3/7/2022	-1
3/30/2022	-1
4/5/2022	-1

date	in_months
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

Usando `3` como argumento `first_month_of_year` na função `inmonths()`, a função começa o ano em 1º de março. A função `inmonths()` então divide o ano em trimestres: Mar-May, Jun-Aug, Sep-Nov, Dec-Feb. Portanto, 15 de maio cai no primeiro trimestre do ano (março-maio).

Diagrama da função `inmonths()` com março definido como primeiro mês do ano



Qualquer transação que ocorra nesses meses retornará um resultado booleano de TRUE.

Exemplo 4: Exemplo de objeto de gráfico

Script de carregamento e expressão de gráfico

Visão geral

São usados o mesmo conjunto de dados e cenário do primeiro exemplo.

No entanto, neste exemplo, o conjunto de dados permanece inalterado e é carregado no aplicativo. O cálculo que determina se as transações ocorreram no mesmo trimestre de 15 de maio de 2022 é criado como uma medida em um gráfico no aplicativo.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão:

- `date`

Para calcular se as transações ocorreram no mesmo trimestre de 15 de maio, crie a seguinte medida:

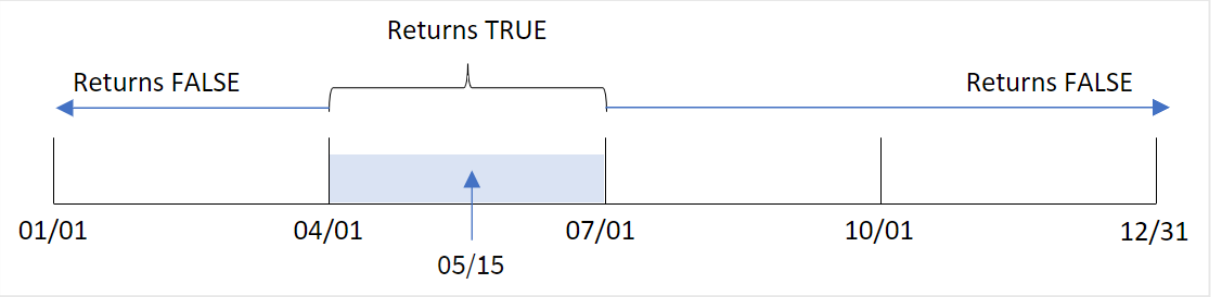
```
=inmonths(3,date,'05/15/2022', 0)
```

Tabela de resultados

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

O campo “in_months” é criado no gráfico usando a função inmonths(). O primeiro argumento fornecido é 3, que divide o ano em segmentos trimestrais. O segundo argumento identifica qual campo está sendo avaliado, o campo de data neste exemplo. O terceiro argumento é uma data codificada para 15 de maio, que é base_date, e period_no de 0 é o argumento final.

Diagrama da função `inmonths()` com segmentos trimestrais



Maio cai no segundo trimestre do ano. Portanto, qualquer transação que ocorra entre 1º de abril e 30 de junho retornará um resultado booleano de TRUE. Isso é validado na tabela de resultados.

Exemplo 5: Cenário

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados que é carregado em uma tabela denominada "`Products`".
- A tabela contém os seguintes campos:
    - product ID
    - product type
    - manufacture date
    - cost price

O usuário final deseja um gráfico que mostre, por tipo de produto, o custo dos produtos fabricados no primeiro segmento de 2021. O usuário gostaria de poder definir o comprimento desse segmento.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta.

No início do script de carregamento, é criada uma variável `vPeriod` que é vinculada ao controle de entrada da variável.

Faça o seguinte:

1. No painel de ativos, clique em **Objetos personalizados**.
2. Selecione o **pacote Qlik Dashboard**, crie um objeto de **Entrada de variável**.
3. Insira um título para o objeto de gráfico.
4. Em **Variável**, selecione **vPeriod** como o nome e defina o objeto para ser exibido como um **Menu suspenso**.
5. Em **Valores**, clique em valores **Dinâmicos**. Insira o seguinte: `='1~month|2~bi-month|3~quarter|4~tertial|6~half-year'.`
6. Adicione uma nova tabela à pasta.
7. Em **Dados** no painel de propriedades, adicione `product_type` como uma dimensão.
8. Adicione a seguinte expressão como medida: `=sum(if(inmonths($(vPeriod),manufacture_date,makedate(2022,01,01),0),cost_price,0))`
9. Defina o **Formato numérico** da medida como **Dinheiro**.

Tabela de resultados

product_type	**=sum(if(inmonths($(vPeriod),manufacture_date,makedate(2022,01,01),0),cost_price,0))**
produto A	$88.27
produto B	$37.23
produto C	$17.17
produto D	$0.00

A função `inmonths()` usa a entrada do usuário como argumento para definir o tamanho do segmento inicial do ano. A função transmite a data de fabricação de cada um dos produtos como o segundo argumento da função `inmonths()`. Ao usar 1º de janeiro como terceiro argumento na função `inmonths()`, os produtos com datas de fabricação que caem no segmento de abertura do ano retornarão um valor booleano de TRUE e, portanto, a função sum adicionará os custos desses produtos.

inmonthstodate

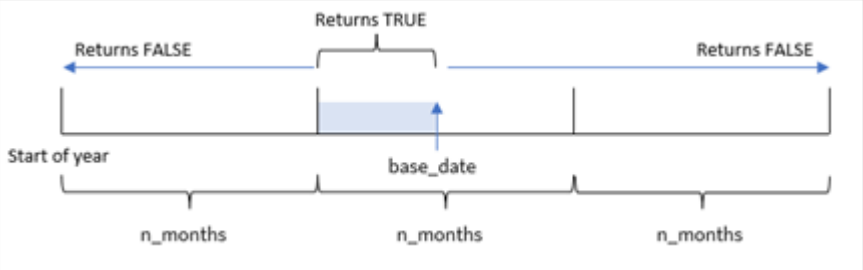
Essa função descobre se um carimbo de data/hora está dentro da parte de um período do mês, bimestre, trimestre, quadrimestre ou semestre, até e incluindo o último milissegundo de `base_date`. Também é possível descobrir se uma data/hora está dentro de um período de tempo anterior ou seguinte.

Sintaxe:

```
InMonths (n_months, timestamp, base_date, period_no[, first_month_of_year ])
```

Tipo de dados de retorno: Booleano

Diagrama da função `inmonthstodate`.



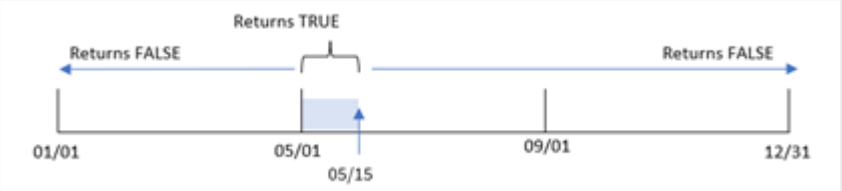
Argumentos

Argumento	Descrição
n_months	O número de meses que define o período. Um inteiro ou uma expressão que se resolve como um inteiro que deve ser um dos seguintes: 1 (equivalente à função inmonth()), 2 (bimestre), 3 (equivalente à função inquarter()), 4 (quadrimestre) ou 6 (semestre).
timestamp	A data que você deseja comparar com **base_date**.
base_date	Data que é usada para avaliar o período.
period_no	O período pode ser deslocado por **period_no**, um inteiro ou expressão que resolve um inteiro, no qual o valor 0 indica o período que contém **base_date**. Valores negativos em **period_no** indicam períodos precedentes e valores positivos indicam períodos sucessivos.
first_month_of_year	Se desejar trabalhar com anos (fiscais) que não comecem em janeiro, indique um valor entre 2 e 12 em **first_month_of_year**.

Na função `inmonthstodate()`, `base_date` atua como o ponto final do segmento do ano específico do qual ele faz parte.

Por exemplo, se o ano fosse dividido em segmentos terciais e `base_date` fosse 15 de maio, qualquer carimbo de data/hora entre o início de janeiro e o final de abril retornaria um resultado booleano de FALSE. As datas entre 1º de maio e 15 de maio retornariam TRUE. O resto do ano retornaria FALSE.

Diagrama do intervalo de resultados booleanos da função `inmonthstodate`.



Os seguintes segmentos do ano estão disponíveis na função como argumentos `n_month`.

Argumentos `n_month`

Período	Número de meses
mês	1
bimestre	2
trimestre	3
tercil	4
semestre	6

Quando usar

A função `inmonthstodate()` retorna um resultado booleano. Normalmente, esse tipo de função é usado como condição em `if expression`. Usando a função `inmonthstodate()`, você pode selecionar o período que deseja avaliar. Por exemplo, fornecer uma variável de entrada que permite ao usuário identificar os produtos fabricados no mês, trimestre ou semestre de um período, até uma determinada data.

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplos de funções

Exemplo	Resultado
`inmonthstodate(4, '01/25/2013', '04/25/2013', 0)`	Retorna True porque o valor de timestamp, 01/25/2013, está dentro do período de quatro meses 01/01/2013 até o final de 04/25/2013, em que o valor de base_date, 04/25/2013, está.
`inmonthstodate(4, '04/26/2013', '04/25/2006', 0)`	Retorna False, porque 04/26/2013 está fora do mesmo período do exemplo anterior.
`inmonthstodate(4, '09/25/2005', '02/01/2006', -1)`	Retorna True, porque o valor de period_no, -1, desloca o período de pesquisa para um período de quatro meses (o valor de n-months), o que faz com que o período de pesquisa seja de 01/09/2005 a 02/01/2006.
`inmonthstodate(4, '04/25/2006', '06/01/2006', 0, 3)`	Retorna True, porque o valor de first_month_of_year está definido como 3, o que faz com que o período de pesquisa seja de 03/01/2006 a 06/01/2006 em vez de 05/01/2006 a 06/01/2006.

Exemplo 1: Sem argumentos adicionais

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações para 2022, que é carregado em uma tabela denominada "`Transactions`".
- Um campo de data na variável de sistema `DateFormat`, no formato `(MM/DD/YYYY)`.
- Uma instrução de carregamento anterior contendo:
    - A função `inmonthstodate()` que é definida como o campo, "`in_months_to_date`". Isso determina quais transações ocorreram no trimestre até 15 de maio de 2022.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `in_months_to_date`

Tabela de resultados

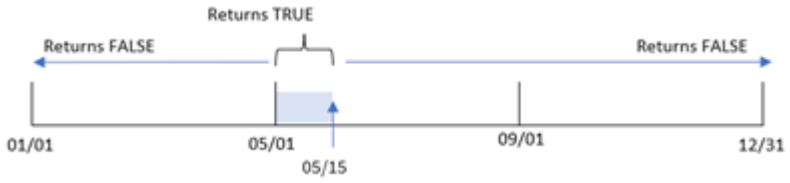
date	in_months_to_date
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	0
4/1/2022	-1
5/7/2022	-1
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0

date	in_months_to_date
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

O campo "in_months_to_date" é criado na instrução de carregamento anterior usando a função inmonthstodate().

O primeiro argumento fornecido é 3, que divide o ano em segmentos trimestrais. O segundo argumento identifica qual campo está sendo avaliado. O terceiro argumento é uma data codificada para 15 de maio, que é a base_date que define o limite final do segmento. Um period_no de 0 é o argumento final.

Diagrama da função inmonthstodate sem argumentos adicionais.



Qualquer transação que ocorra entre 1º de abril e 15 de maio retorna um resultado booleano de TRUE. As datas de transação fora desse período retornam FALSE.

Exemplo 2: period_no

Script de carregamento e resultados

Visão geral

São usados o mesmo conjunto de dados e cenário do primeiro exemplo.

No entanto, neste exemplo, a tarefa é criar um campo, “previous_qtr_to_date”, que determina se as transações ocorreram um trimestre antes de 15 de maio.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `previous_qtr_to_date`

Tabela de resultados

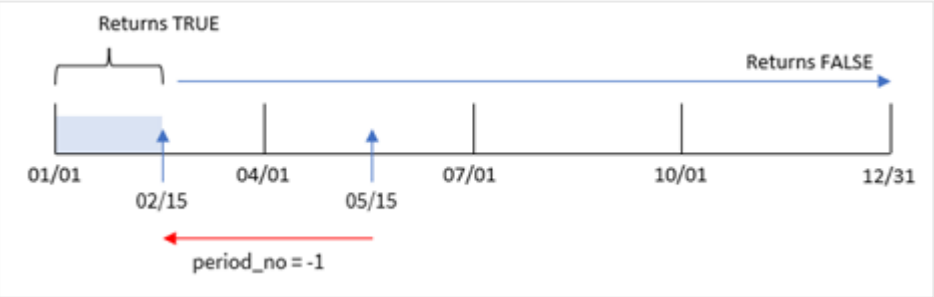
date	previous_qtr_to_date
1/7/2022	-1
1/19/2022	-1
2/5/2022	-1
2/28/2022	0
3/16/2022	0
4/1/2022	0
5/7/2022	0
5/16/2022	0

date	previous_qtr_to_date
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

Ao usar `-1` como o argumento `period_no` na função `inmonthstodate()`, a função muda os limites do segmento do ano comparador em um trimestre.

15 de maio cai no segundo trimestre do ano e, portanto, o segmento inicialmente equivale a um período entre 1º de abril e 15 de maio. O argumento `period_no` compensa esse segmento em menos três meses. Os limites da data passam de 1º de janeiro para 15 de fevereiro.

Diagrama da função `inmonthstodate` com period_no definido como -1.



Portanto, qualquer transação que ocorra entre 1º de janeiro e 15 de fevereiro retornará um resultado booleano de TRUE.

Exemplo 3: first_month_of_year

Script de carregamento e resultados

Visão geral

São usados o mesmo conjunto de dados e cenário do primeiro exemplo.

Neste exemplo, a política organizacional é que março seja o primeiro mês do exercício financeiro.

Crie um campo, "in_months_to_date", que determina quais transações ocorreram no mesmo trimestre até 15 de maio de 2022.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- date
- in_months_to_date

Tabela de resultados

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

Usando `3` como argumento `first_month_of_year` na função `inmonthstodate()`, a função começa o ano em 1º de março e depois o divide em trimestres com base no primeiro argumento fornecido. Portanto, os segmentos de trimestre são:

- Mar-May
- Jun-Aug
- Sep-Nov
- Dec-Feb

O `base_date` de 15 de maio então segmenta o trimestre de março a maio, definindo seu limite final como 15 de maio.

Diagrama da função `inmonthstodate` com março definido como primeiro mês do ano.

Portanto, qualquer transação que ocorra entre 1º de março e 15 de maio retornará um resultado booleano de TRUE, e transações com datas fora desses limites retornarão um valor de FALSE.

Exemplo 4 – Exemplo de gráfico

Script de carregamento e expressão de gráfico

Visão geral

São usados o mesmo conjunto de dados e cenário do primeiro exemplo.

Neste exemplo, o conjunto de dados permanece inalterado e carregado no aplicativo. A tarefa é criar um cálculo que determine se as transações ocorreram no mesmo trimestre de 15 de maio como medida em um gráfico do aplicativo.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão:

```
date
```

Para calcular se as transações ocorreram no mesmo trimestre de 15 de maio, crie a seguinte medida:

```
=inmonthstodate(3,date,'05/15/2022', 0)
```

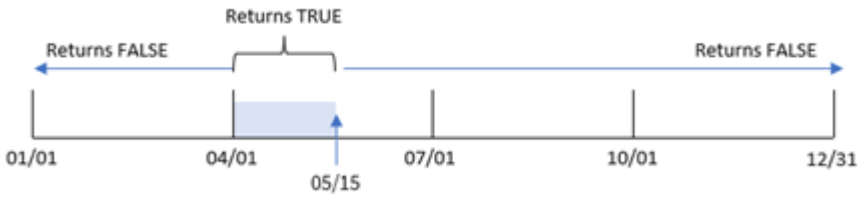
Tabela de resultados

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
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

A medida "`'in_months_to_date`" é criada no gráfico usando a função `inmonthstodate()`.

O primeiro argumento fornecido é 3, que divide o ano em segmentos trimestrais. O segundo argumento identifica qual campo está sendo avaliado. O terceiro argumento é uma data codificada de 15 de maio, base_date que define o limite final do segmento. Um period_no de 0 é o argumento final.

Diagrama da função inmonthstodate com segmentos trimestrais.



Qualquer transação que ocorra entre 1º de abril e 15 de maio retornará um resultado booleano de TRUE. As datas de transação fora desse segmento retornarão FALSE.

Exemplo 5: Cenário

Script de carregamento e resultados

Visão geral

Neste exemplo, um conjunto de dados é carregado em uma tabela chamada "Sales". A tabela contém os seguintes campos:

- ID do produto
- Tipo de produto
- Data de venda
- Preço de venda

O usuário final deseja um gráfico que mostre, por tipo de produto, as vendas dos produtos vendidos no período que antecede 24 de dezembro de 2022. O usuário gostaria de poder definir a duração desse período.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta.

No início do script de carregamento, é criada uma variável `vPeriod` que é vinculada ao controle de entrada da variável.

Faça o seguinte:

1. No painel de ativos, clique em **Objetos personalizados**.
2. Selecione o **pacote Qlik Dashboard** e adicione uma **Entrada de variável** à sua pasta.
3. Insira um título para o gráfico.
4. Em **Variável**, selecione **vPeriod** como o nome e defina o objeto a ser exibido como um **Menu suspenso**.
5. Em **Valores**, clique em valores **Dinâmicos**. Insira o seguinte:
   `='1~month|2~bi-month|3~quarter|4~tertial|6~half-year'.`
6. Adicione uma nova tabela à pasta.
7. Em **Dados** no painel de propriedades, adicione `product_type` como uma dimensão.
8. Adicione a seguinte expressão como medida:
   `=sum(if(inmonthstodate($(vPeriod),sales_date,makedate(2022,12,24),0),sales_price,0))`
9. Defina o **Formato numérico** da medida como **Dinheiro**.

Tabela de resultados

product_type	**=sum(if(inmonthstodate($(vPeriod),sales_date,makedate(2022,12,24),0),sales_price,0))**
produto A	$186.48
produto B	$190.52
produto C	$220.43
produto D	$261.46

A função `inmonthstodate()` usa a entrada do usuário como argumento para definir o tamanho do segmento inicial do ano.

A função transmite a data de venda de cada um dos produtos como o segundo argumento da função `inmonthstodate()`. Ao usar 24 de dezembro como terceiro argumento na função `inmonthstodate()`, os produtos com datas de venda que ocorrem no período definido até 24 de dezembro, inclusive, retornam um valor booleano de TRUE. A função sum adiciona as vendas desses produtos.

inmonthtodate

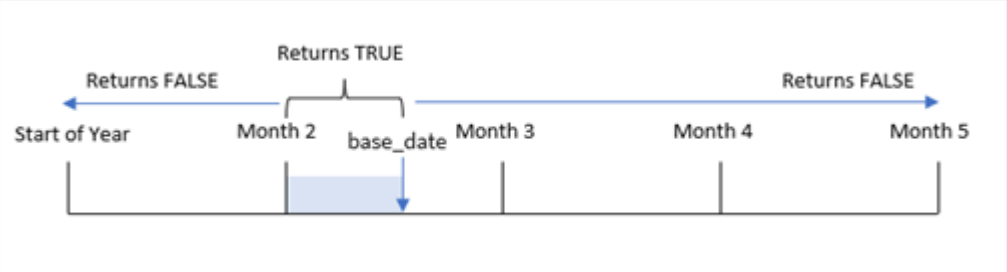
Retornará True se **date** estiver na parte do mês que contém a **basedate** até e inclusive o último milissegundo da **basedate**.

Sintaxe:

```
InMonthToDate (timestamp, base_date, period_no)
```

Tipo de dados de retorno: Booleano

Diagrama da função `inmonthtodate`.



A função `inmonthtodate()` identifica um mês selecionado como um segmento. O limite inicial é o início do mês. O limite final pode ser definido como uma data posterior no mês. Em seguida, ela determina se um conjunto de datas cai nesse segmento ou não, retornando um valor booleano de TRUE ou FALSE.

Argumentos

Argumento	Descrição
timestamp	A data que você deseja comparar com **base_date**.
base_date	Data que é usada para avaliar o mês.
period_no	O mês pode ser deslocado por **period_no**. **period_no** é um inteiro, em que o valor 0 indica o mês que contém **base_date**. Valores negativos em **period_no** indicam meses precedentes e valores positivos indicam meses sucessivos.

Quando usar

A função `inmonthtodate()` retorna um resultado booleano. Normalmente, esse tipo de função é usado como condição em `if expression`. A função `inmonthtodate()` retorna uma agregação ou cálculo que depende se uma data ocorreu no mês até e incluindo a data em questão.

Por exemplo, a função `inmonthtodate()` pode ser usada para identificar todos os equipamentos fabricados em um mês até uma data específica.

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplos de funções

Exemplo	**Resultado**
`inmonthtodate ('01/25/2013', '25/01/2013', 0)`	Retorna True
`inmonthtodate ('01/25/2013', '24/01/2013', 0)`	Retorna False
`inmonthtodate ('01/25/2013', '28/02/2013', -1)`	Retorna True

Exemplo 1: Sem argumentos adicionais

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações para 2022 é carregado em uma tabela denominada "`Transactions`".
- Um campo de data é fornecido no formato da `variável` de sistema DateFormat (`MM/DD/YYYY`)
- Uma instrução de carregamento anterior contendo:
    - A função `inmonthtodate()` que é definida como o campo, "`in_month_to_date`". Isso determina quais transações ocorreram entre 1º de julho e 26 de julho de 2022.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `in_month_to_date`

Tabela de resultados

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

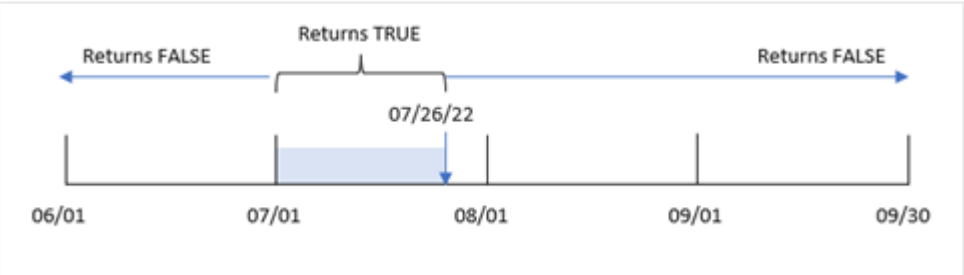
date	in_month_to_date
6/15/2022	0
6/26/2022	0
7/9/2022	-1
7/22/2022	-1
7/23/2022	-1
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

O campo "`in_month_to_date`" é criado na instrução de carregamento anterior usando a função `inmonthtodate()`.

O primeiro argumento identifica qual campo está sendo avaliado. O segundo argumento é uma data codificada, 26 de julho, que é a `base_date`. Esse argumento `base_date` identifica qual mês é segmentado e o limite final desse segmento.

Um `period_no` de `0` é o argumento final que significa que a função não está comparando meses anteriores ou posteriores ao mês segmentado.

Diagrama da função `inmonthtodate` sem argumentos adicionais.



Como resultado, qualquer transação que ocorra entre 1º de julho e 26 de julho retorna um resultado booleano de TRUE. Qualquer transação que ocorra em julho após 26 de julho retorna um resultado booleano de FALSE, assim como qualquer transação em qualquer outro mês do ano.

Exemplo 2: period_no

Script de carregamento e resultados

Visão geral

São usados o mesmo conjunto de dados e cenário do primeiro exemplo.

Neste exemplo, a tarefa é criar um campo, “six_months_prior”, que determina quais transações ocorreram seis meses antes de 1º de julho e 26 de julho.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

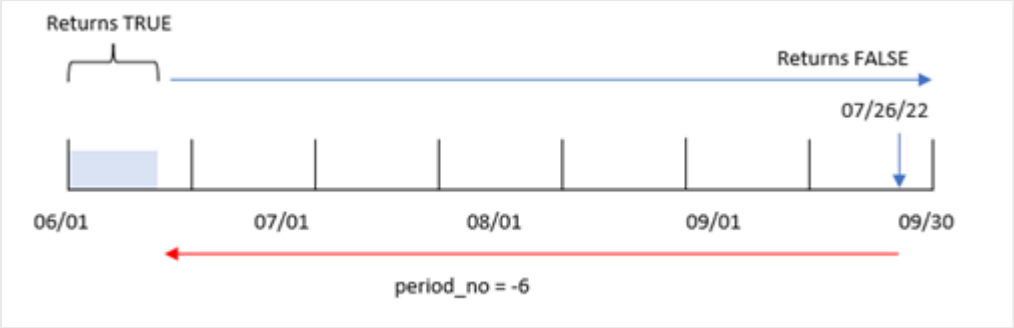
- date
- six_months_prior

Tabela de resultados

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
9/26/2022	0
10/14/2022	0
10/29/2022	0

Usando `-6` como o argumento `period_no` na função `inmonthtodate()`, os limites do segmento do mês comparador mudam em seis meses. Inicialmente, o segmento do mês equivale a um período entre 1º de julho e 26 de julho. Em seguida, `period_no` desloca esse segmento em menos seis meses, e os limites de data são alterados e caem entre 1º de janeiro e 26 de janeiro.

Diagrama da função `inmonthtodate` com period_no definido como -6.



Como resultado, qualquer transação que ocorra entre 1º de janeiro e 26 de janeiro retornará um resultado booleano de TRUE.

Exemplo 3: Exemplo de gráfico

Script de carregamento e expressão de gráfico

Visão geral

São usados o mesmo conjunto de dados e cenário do primeiro exemplo.

Neste exemplo, o conjunto de dados permanece inalterado e carregado no aplicativo. A tarefa é criar um cálculo que determine se as transações ocorreram entre 1º e 26 de julho como uma medida em um gráfico do aplicativo.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão:

```
date
```

Para calcular se as transações ocorreram entre 1º de julho e 26 de julho, crie a seguinte medida:

```
=inmonthtodate(date,'07/26/2022', 0)
```

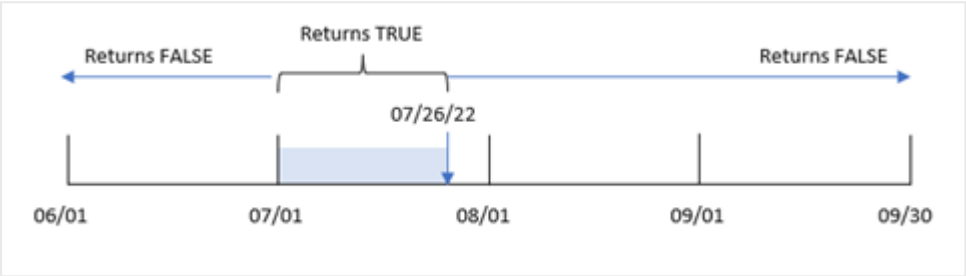
Tabela de resultados

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
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

A medida do campo "in_month_to_date" é criada no gráfico usando a função inmonthtodate().

O primeiro argumento identifica qual campo está sendo avaliado. O segundo argumento é uma data codificada, 26 de julho, que é a base_date. Esse argumento base_date identifica qual mês é segmentado e o limite final desse segmento. Um period_no de 0 é o argumento final. Isso significa que a função não está comparando os meses anteriores ou posteriores ao mês segmentado.

Diagrama da função inmonthtodate sem argumentos adicionais.



Como resultado, qualquer transação que ocorra entre 1º de julho e 26 de julho retorna um resultado booleano de TRUE. Qualquer transação que ocorra em julho após 26 de julho retorna um resultado booleano de FALSE, assim como qualquer transação em qualquer outro mês do ano.

Exemplo 4: Cenário

Script de carregamento e resultados

Visão geral

Neste exemplo, um conjunto de dados é carregado em uma tabela chamada "Products". A tabela contém os seguintes campos:

- ID do produto
- Data de fabricação
- Preço de custo

Devido a erros de equipamentos, os produtos fabricados no mês de julho de 2022 estavam com defeito. O problema foi resolvido em 27 de julho de 2022.

O usuário final deseja um gráfico que mostre, por mês, o status dos produtos que foram fabricados como "com defeito" (booleano TRUE) ou "sem defeito" (booleano FALSE) e também o custo dos produtos fabricados naquele mês.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `=monthname(manufacture_date)`
- `=if(Inmonthtodate(manufacture_date,makedate(2022,07,26),0),'Defective','Faultless')`

Para calcular o custo da soma dos produtos, crie esta medida:

```
=sum(cost_price)
```

Defina o **Formato numérico** da medida como **Dinheiro**.

Tabela de resultados

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

monthname (manufacture_date)	if(Inmonthtodate(manufacture_date,makedate (2022,07,26),0),'Defective','Faultless')	Sum(cost_ price)
Sep 2022	Faultless	$84.21
Oct 2022	Faultless	$163.91

A função `inmonthtodate()` retorna um valor booleano ao avaliar as datas de fabricação de cada um dos produtos.

Para as datas que retornam um valor booleano de TRUE, o produto é marcado como “Defective”. Para qualquer produto que retorne um valor de FALSE e, portanto, não seja fabricado no mês até 26 de julho, inclusive, ela marca os produtos como “Faultless”.

inquarter

Esta função retornará True se **timestamp** estiver dentro do trimestre que contém **base_date**.

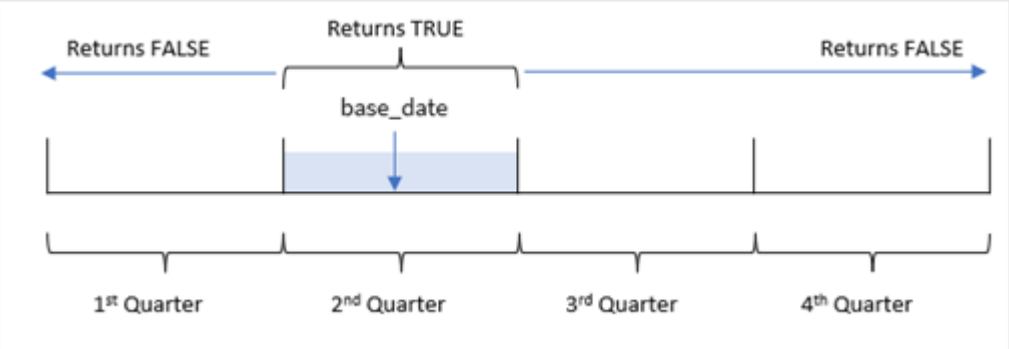
Sintaxe:

```
InQuarter (timestamp, base_date, period_no[, first_month_of_year])
```

Tipo de dados de retorno: Booleano

No Qlik Sense, o valor booleano “true” é representado por -1, e o valor falso é representado por 0.

Diagrama do intervalo da função `inquarter()`



Em outras palavras, a função `inquarter()` divide o ano em quatro trimestres iguais entre 1º de janeiro e 31 de dezembro. Você pode usar o argumento `first_month_of_year` para alterar qual mês é considerado o primeiro em seu aplicativo, e os trimestres mudarão com base nesse argumento. A função `base_date` identifica qual trimestre deve ser usado como comparador da função. Finalmente, a função retorna um resultado booleano ao comparar valores de data com esse segmento trimestral.

Quando usar

A função `inquarter()` retorna um resultado booleano. Normalmente, esse tipo de função será usado como uma condição em um `if expression`. Isso retorna uma agregação ou cálculo que depende se uma data ocorreu no trimestre selecionado.

Por exemplo, a função `inquarter()` pode ser usada para identificar todos os equipamentos fabricados em um segmento trimestral com base nas datas em que o equipamento foi fabricado.

Argumentos

Argumento	Descrição
timestamp	A data que você deseja comparar com **base_date**.
base_date	Data que é usada para avaliar o trimestre.
period_no	O trimestre pode ser deslocado por **period_no**. **period_no** é um inteiro, em que o valor 0 indica o trimestre que contém **base_date**. Valores negativos em **period_no** indicam trimestres precedentes e valores positivos indicam trimestres sucessivos.
first_month_of_year	Se desejar trabalhar com anos (fiscais) que não comecem em janeiro, indique um valor entre 2 e 12 em **first_month_of_year**.

Você pode usar os seguintes valores para definir o primeiro mês do ano no argumento `first_month_of_year`:

Valores first_month_of_year

Month	Valor
Fevereiro	2
Março	3
Abril	4
Maio	5
Junho	6
Julho	7
Agosto	8
Setembro	9
Outubro	10
Novembro	11
Dezembro	12

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplos de funções

Exemplo	**Resultado**
inquarter ('01/25/2013', '01/01/2013', 0)	Retorna TRUE
inquarter ('01/25/2013', '04/01/2013', 0)	Retorna FALSE
inquarter ('01/25/2013', '01/01/2013', -1)	Retorna FALSE
inquarter ('12/25/2012', '01/01/2013', -1)	Retorna TRUE
inquarter ('01/25/2013', '03/01/2013', 0, 3)	Retorna FALSE
inquarter ('03/25/2013', '03/01/2013', 0, 3)	Retorna TRUE

Exemplo 1: Nenhum argumento adicional

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações em 2022 que é carregado em uma tabela denominada "`Transactions`".
- Um carregamento anterior que contém a função `inquarter()` definida como o `campo` “in_quarter” e determina quais transações ocorreram no mesmo trimestre de 15 de maio de 2022.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `in_quarter`

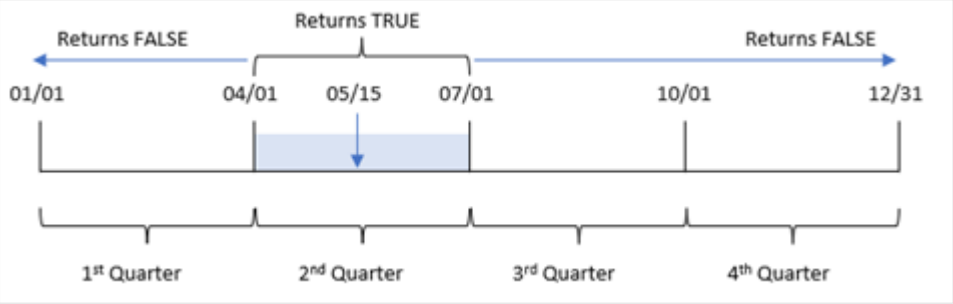
Tabela de resultados

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

date	in_quarter
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

O campo "in_quarter" é criado na instrução de carregamento anterior usando a função `inquarter()`. O primeiro argumento identifica qual campo está sendo avaliado. O segundo argumento é uma data codificada para 15 de maio que identifica qual trimestre definir como o comparador. Um `period_no` de `0` é o argumento final e garante que a função `inquarter()` não compare trimestres anteriores ou posteriores ao trimestre segmentado.

Diagrama da função `inquarter()` com 15 de maio como data base



Qualquer transação que ocorra entre 1º de abril e o final de 30 de junho retorna um resultado booleano de TRUE.

Exemplo 2: period_no

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações em 2022 que é carregado em uma tabela denominada "`Transactions`".
- Um carregamento anterior que contém a função `inquarter()` definida como o campo "`previous_quarter`" e determina quais transações ocorreram no trimestre anterior ao trimestre de 15 de maio de 2022.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `previous_qtr`

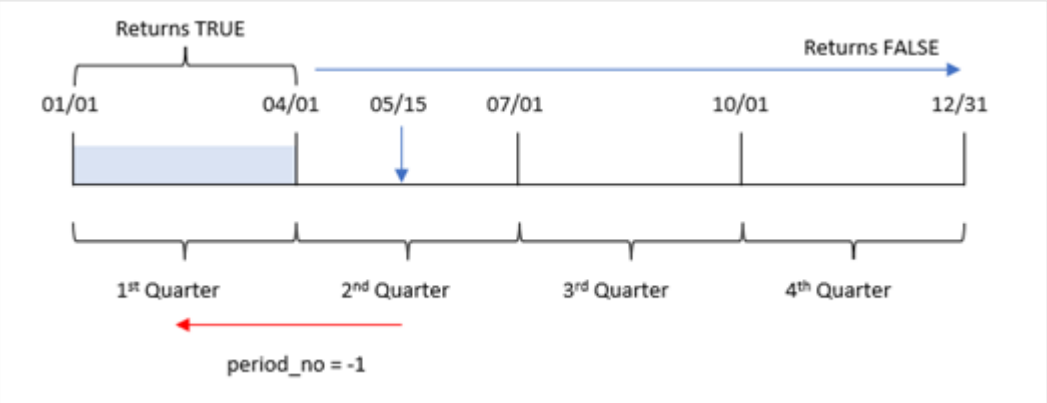
Tabela de resultados

date	previous_qtr
1/7/2022	-1
1/19/2022	-1
2/5/2022	-1
2/28/2022	-1

date	previous_qtr
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
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Usar `-1` como o argumento `period_no` na função `inquarter()` desloca os limites do trimestre comparador para trás em um trimestre inteiro. 15 de maio cai no segundo trimestre do ano e, portanto, o segmento inicialmente equivale ao trimestre de 1º de abril a 30 de junho. `period_no` desloca esse segmento em três meses negativos e faz com que os limites de data se tornem de 1º de janeiro a 30 de março.

Diagrama da função `inquarter()` com 15 de maio como data base



Portanto, qualquer transação que ocorra entre 1º de janeiro e 30 de março retornará um resultado booleano de TRUE.

Exemplo 3: first_month_of_year

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações em 2022 que é carregado em uma tabela denominada "`Transactions`".
- Um carregamento anterior que contém a função `inquarter()` definida como o campo "`in_quarter`" e determina quais transações ocorreram no mesmo trimestre de 15 de maio de 2022.

No entanto, neste exemplo, a política organizacional é que março seja o primeiro mês do exercício financeiro.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

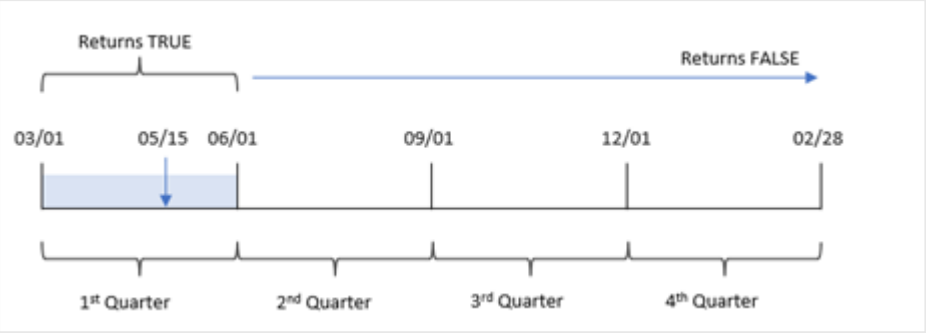
- `date`
- `previous_qtr`

Tabela de resultados

date	previous_qtr
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	-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

Usar `3` como argumento `first_month_of_year` na função `inquarter()` define 1º de março como o início do ano e depois divide o ano em trimestres. Portanto, os segmentos trimestrais são de março a maio, junho a agosto, setembro a novembro, dezembro a fevereiro. A `base_date` de 15 de maio define o trimestre de março a maio como o trimestre comparador da função.

Diagrama da função `inquarter()` com março definido como o primeiro mês do ano



Portanto, qualquer transação que ocorra entre 1º de março e 31 de maio retornará um resultado booleano de TRUE.

Exemplo 4: Exemplo de objeto de gráfico

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações em 2022 que é carregado em uma tabela denominada "`Transactions`".
- Um carregamento anterior que contém a função `inquarter()` definida como o campo "`in_quarter`" e determina quais transações ocorreram no mesmo trimestre de 15 de maio de 2022.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão:

- `date`

Crie a medida a seguir para calcular se as transações ocorreram no mesmo trimestre de 15 de maio:

```
=inquarter(date,'05/15/2022', 0)
```

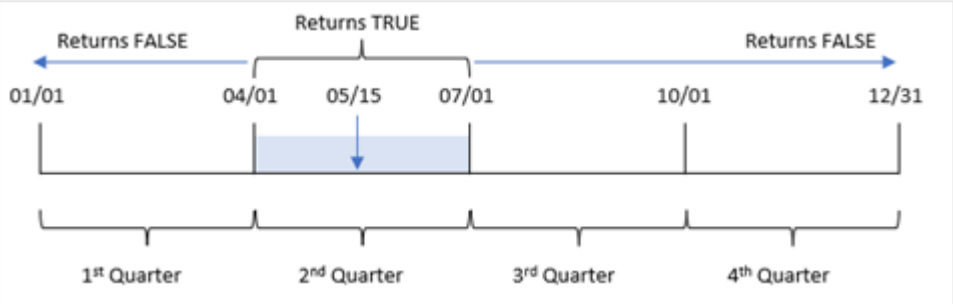
Tabela de resultados

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

date	in_quarter
10/14/2022	0
10/29/2022	0

A medida "in_quarter" é criada no gráfico usando a função `inquarter()`. O primeiro argumento identifica qual campo está sendo avaliado. O segundo argumento é uma data codificada para 15 de maio que identifica qual trimestre definir como o comparador. Um `period_no` de 0 é o argumento final e garante que a função `inquarter()` não compare trimestres anteriores ou posteriores ao trimestre segmentado.

Diagrama da função `inquarter()` com 15 de maio como data base



Qualquer transação que ocorra entre 1º de abril e o final de 30 de junho retorna um resultado booleano de TRUE.

Exemplo 5: Cenário

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados que é carregado em uma tabela denominada "`Products`".
- A tabela contém os seguintes campos:
  - product ID
  - product type
  - manufacture date
  - cost price

Foi identificado que, devido a um erro no equipamento, os produtos que foram fabricados no trimestre de 15 de maio de 2022 estavam com defeito. O usuário final deseja um gráfico que mostre, por nome do trimestre, o status de quais produtos fabricados estavam "com defeito" ou "sem defeito", bem como o custo dos produtos fabricados naquele trimestre.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão:

```
=quartername(manufacture_date)
```

Crie as seguintes medidas:

- `=if(only(InQuarter(manufacture_date,makedate(2022,05,15),0)),'Defective','Faultless')`, para identificar quais produtos estão com defeito e quais estão sem defeito usando a função `inquarter()`.
- `=sum(cost_price)`, para mostrar a soma do custo de cada produto.

Faça o seguinte:

1. Defina o **Formato numérico** da medida como **Dinheiro**.
2. Em **Aparência**, desative **Totais**.

Tabela de resultados

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

A função `inquarter()` retorna um valor booleano ao avaliar as datas de fabricação de cada um dos produtos. Para qualquer produto fabricado no trimestre que contenha 15 de maio, a função retorna `inquarter()` um valor booleano de TRUE e marca os produtos como “Defective”. Para qualquer produto que retorne um valor de FALSE e, portanto, não fabricado naquele trimestre, ela marca os produtos como “Faultless”.

inquartertodate

Esta função retornará True se **timestamp** estiver na parte do trimestre que contém a **base_date** até e inclusive o último milissegundo de **base_date**.

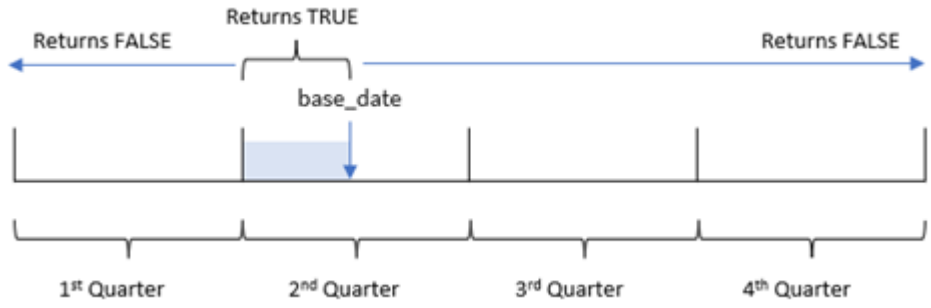
Sintaxe:

```
InQuarterToDate (timestamp, base_date, period_no [, first_month_of_year])
```

Tipo de dados de retorno: Booleano

> *No Qlik Sense, o valor booleano “true” é representado por -1, e o valor falso é representado por 0.*

Diagrama da função inquartertodate



A função `inquartertodate()` divide o ano em quatro trimestres iguais entre 1º de janeiro e 31 de dezembro (ou o início do ano definido pelo usuário e sua data final correspondente). Usando a `base_date`, a função segmentará um trimestre específico, com `base_date` identificando o trimestre e a data máxima permitida para esse segmento de trimestre. Por fim, a função retorna um resultado booleano ao comparar os valores de data prescritos com esse segmento.

Argumentos

Argumento	Descrição
timestamp	A data que você deseja comparar com **base_date**.
base_date	Data que é usada para avaliar o trimestre.
period_no	O trimestre pode ser deslocado por **period_no**. **period_no** é um inteiro, em que o valor 0 indica o trimestre que contém **base_date**. Valores negativos em **period_no** indicam trimestres precedentes e valores positivos indicam trimestres sucessivos.
first_month_of_year	Se desejar trabalhar com anos (fiscais) que não comecem em janeiro, indique um valor entre 2 e 12 em **first_month_of_year**.

Quando usar

A função `inquartertodate()` retorna um resultado booleano. Normalmente, esse tipo de função será usado como uma condição em uma expressão `if`. A função `inquartertodate()` seria usada para retornar uma agregação ou um cálculo, dependendo de se uma data avaliada ocorreu no trimestre, até e incluindo a data em questão.

Por exemplo, a função `inquartertodate()` pode ser usada para identificar todos os equipamentos fabricados em um trimestre até uma data específica.

Exemplos de funções

Exemplo	Resultado
`inquartertodate ('01/25/2013', '03/25/2013', 0)`	Retorna `TRUE`, já que o valor de `timestamp`, 25/01/2013, está dentro do período de três meses de 01/01/2013 a 25/03/2013, no qual se encontra o valor de `base_date`, 25/03/2013.
`inquartertodate ('04/26/2013', '03/25/2013', 0)`	Retorna `FALSE`, já que 26/04/2013 está fora do mesmo período do exemplo anterior.
`inquartertodate ('02/25/2013', '06/09/2013', -1)`	Retorna `TRUE`, já que o valor de `period_no`, `-1`, desloca o período de pesquisa para um período de três meses (um quarto do ano). Isso torna o período de pesquisa de 01/01/2013 a 09/03/2013.
`inquartertodate ('03/25/2006', '04/15/2006', 0, 2)`	Retorna `TRUE`, já que o valor de `first_month_of_year` está definido como 2, o que torna o período de pesquisa 01/02/2006 a 15/04/2006 em vez de 01/04/2006 a 15/04/2006.

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplo 1: Sem argumentos adicionais

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações para 2022, que é carregado em uma tabela denominada `Transactions`.
- O campo de data fornecido no formato da `variável` de sistema DateFormat (MM/DD/AAAA).
- A criação de um campo, `in_quarter_to_date`, que determina quais transações ocorreram no trimestre até 15 de maio de 2022.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

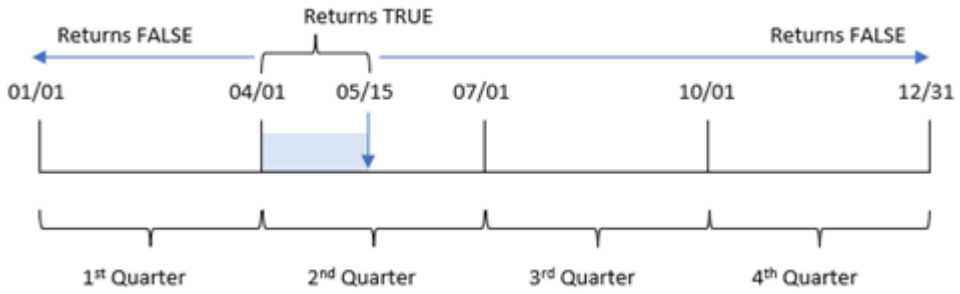
- `date`
- `in_quarter_to_date`

Tabela de resultados

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

O campo `in_quarter_to_date` é criado na instrução de carregamento anterior usando a função `inquartertodate()`. O primeiro argumento fornecido identifica qual campo está sendo avaliado. O segundo argumento é uma data codificada para 15 de maio, que é a `base_date` que identifica qual trimestre segmentar e que define o limite final desse segmento. Um `period_no` de 0 é o argumento final, o que significa que a função não está comparando trimestres anteriores ou posteriores ao trimestre segmentado.

Diagrama da função inquartertodate, sem argumentos adicionais



Qualquer transação que ocorra entre 1º de abril e 15 de maio retorna um resultado booleano de TRUE. As transações com datas de 16 de maio e posteriores retornarão FALSE, assim como qualquer transação antes de 1º de abril.

Exemplo 2: period_no

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- O mesmo conjunto de dados e cenário do primeiro exemplo.
- A criação de um campo, `previous_qtr_to_date`, que determina quais transações ocorreram um trimestre inteiro antes do término do segmento de trimestre em 15 de maio de 2022.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `previous_qtr_to_date`

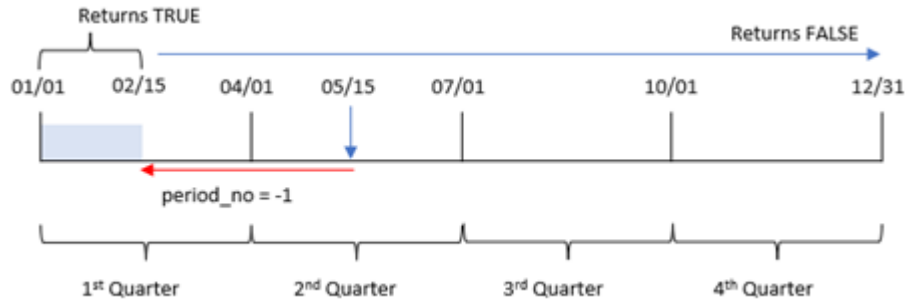
Tabela de resultados

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

date	previous_qtr_to_date
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

Um valor de `period_no` -1 indica que a função `inquartertodate ()` compara o segmento de trimestre de entrada com o trimestre anterior. 15 de maio cai no segundo trimestre do ano e, portanto, o segmento inicialmente equivale a período entre 1º de abril e 15 de maio. `period_no` então desloca esse segmento para três meses antes, fazendo com que os limites de data se tornem de 1º de janeiro a 15 de fevereiro.

Diagrama da função inquartertodate, exemplo de period_no



Portanto, qualquer transação que ocorra entre 1º de janeiro e 15 de fevereiro retornará um resultado booleano de TRUE.

Exemplo 3: first_month_of_year

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- O mesmo conjunto de dados e cenário do primeiro exemplo.
- A criação de um campo, `in_quarter_to_date`, que determina quais transações ocorreram no mesmo trimestre até 15 de maio de 2022.

Neste exemplo, definimos março como o primeiro mês do ano fiscal.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `in_quarter_to_date`

Tabela de resultados

date	in_quarter_to_date
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0

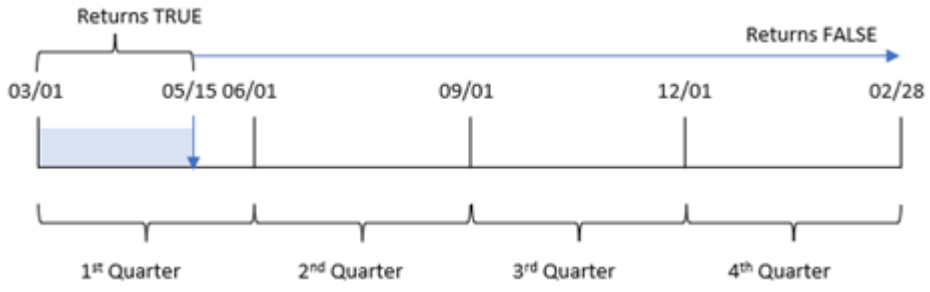
date	in_quarter_to_date
3/16/2022	-1
4/1/2022	-1
5/7/2022	-1
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

Usando 3 como argumento `first_month_of_year` na função `inquartertodate()`, a função começa o ano em 1º de março e, em seguida, divide o ano em trimestres. Portanto, os segmentos de trimestre são:

- Março a maio
- Junho a agosto
- Setembro a novembro
- Dezembro a fevereiro

A `base_date` de 15 de maio então segmenta o trimestre de março a maio, estabelecendo seu limite final como 15 de maio.

Diagrama da função inquartertodate, exemplo com first_month_of_year



Portanto, qualquer transação que ocorra entre 1º de março e 15 de maio retornará um resultado booleano de TRUE, enquanto as transações com datas fora desses limites retornarão um valor de FALSE.

Exemplo 4: Exemplo de objeto de gráfico

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém o mesmo conjunto de dados e cenário do primeiro exemplo. No entanto, neste exemplo, o conjunto de dados inalterado é carregado no aplicativo. O cálculo que determina quais transações ocorreram no mesmo trimestre de 15 de maio é criado como uma medida no objeto de gráfico.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão:`date`.

Crie a seguinte medida:

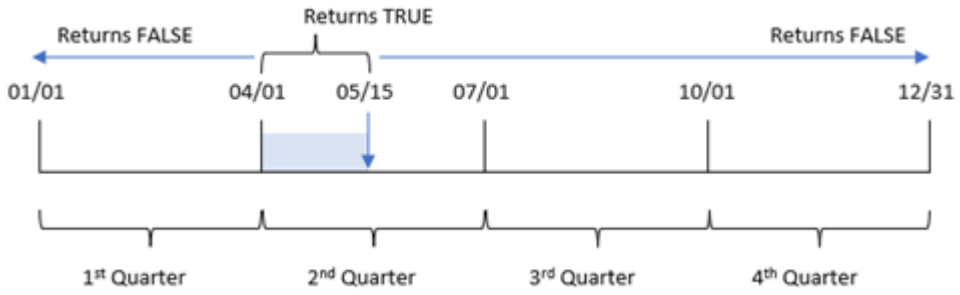
```
=inquartertodate(date,'05/15/2022', 0)
```

Tabela de resultados

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

A medida `in_quarter_to_date` é criada em um objeto de gráfico usando a função `inquartertodate()`. O primeiro argumento é o campo de data que está sendo avaliado. O segundo argumento é uma data codificada para 15 de maio, que é a `base_date` que identifica qual trimestre segmentar e que define o limite final desse segmento. Um `period_no` de 0 é o argumento final, o que significa que a função não está comparando trimestres anteriores ou posteriores ao trimestre segmentado.

Diagrama da função inquartertodate, exemplo de objeto de gráfico



Qualquer transação que ocorra entre 1º de abril e 15 de maio retorna um resultado booleano de TRUE. As transações com datas de 16 de maio e posteriores retornarão FALSE, assim como qualquer transação antes de 1º de abril.

Exemplo 5: Cenário

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados que é carregado em uma tabela denominada `Products`.
- Informações sobre o ID do produto, data de fabricação e preço de custo.

Em 15 de maio de 2022, um erro de equipamento foi identificado no processo de fabricação e resolvido. Os produtos que foram fabricados naquele trimestre até essa data estão com defeito. O usuário final deseja um objeto de gráfico que mostre, por nome de trimestre, o status que indica se o produto está "com defeito" ou "sem defeito", bem como o custo dos produtos fabricados naquele trimestre até o momento.

Script de carregamento

```
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
];
```

Resultados

Faça o seguinte:

1. Carregue os dados e abra uma pasta. Crie uma nova tabela. Crie uma dimensão para mostrar os nomes dos trimestres:
   `=quartername(manufacture_date)`
2. Em seguida, crie uma dimensão para identificar quais dos produtos estão com defeito e quais estão sem defeito:
   `=if(inquartertodate(manufacture_date,makedate(2022,05,15),0),'Defective','Faultless')`
3. Crie uma medida para somar o `cost_price` dos produtos:
   `=sum(cost_price)`
4. Defina o **Formato numérico** da medida como **Dinheiro**.

Tabela de resultados

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

A função `inquartertodate()` retorna um valor booleano ao avaliar as datas de fabricação de cada um dos produtos. Para os casos que retornam um valor booleano de `TRUE`, ela marca os produtos como `'Defective'`. Para qualquer produto que retorne um valor de `FALSE` e, portanto, não tenha sido fabricado no trimestre até e incluindo o dia 15 de maio, ela marca os produtos como `'Faultless'`.

inweek

Essa função retornará True se **timestamp** estiver dentro da semana que contém **base_date**.

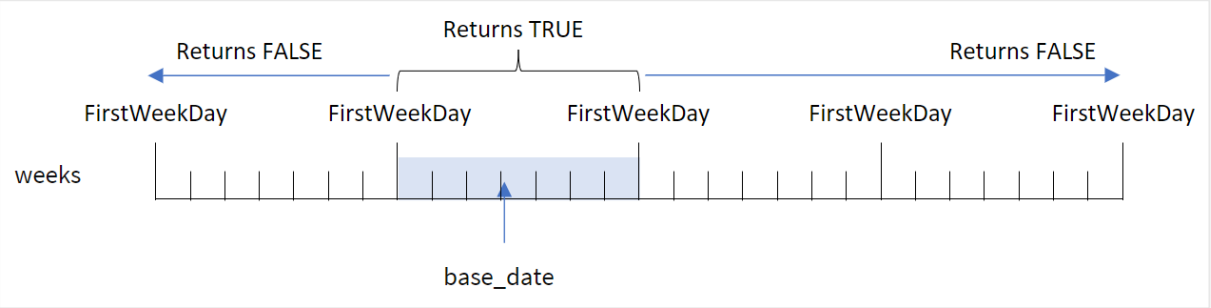
Sintaxe:

```
InWeek (timestamp, base_date, period_no[, first_week_day])
```

Tipo de dados de retorno: Booleano

No Qlik Sense, o valor booleano “true” é representado por -1, e o valor falso é representado por 0.

Diagrama do intervalo da função `inweek()`



A função `inweek()` usa o argumento `base_date` para identificar em qual período de sete dias a data cai. O dia de início da semana é baseado na `variável` de sistema FirstWeekDay. No entanto, você também pode alterar o primeiro dia da semana usando o argumento `first_week_day` da função `inweek()`.

Depois que a semana selecionada for definida, a função retornará resultados booleanos ao comparar os valores de data prescritos com esse segmento da semana.

Quando usar

A função `inweek()` retorna um resultado booleano. Normalmente, esse tipo de função será usado como uma condição em um `if expression`. A função `inweek()` retorna uma agregação ou cálculo que depende se uma data avaliada ocorreu na semana com a data selecionada do argumento `base_date`.

Por exemplo, a função `inweek()` pode ser usada para identificar todos os equipamentos fabricados em uma semana específica.

Argumentos

Argumento	Descrição
timestamp	A data que você deseja comparar com **base_date**.
base_date	Data que é usada para avaliar a semana.
period_no	A semana pode ser deslocada por **period_no**. **period_no** é um inteiro, em que o valor 0 indica a semana que contém **base_date**. Os valores negativos em **period_no** indicam semanas precedentes e os valores positivos indicam semanas subsequentes.
first_week_day	Por padrão, o primeiro dia da semana é domingo (conforme determinado pela variável de sistema FirstWeekDay), começando à meia-noite entre sábado e domingo. O parâmetro **first_week_day** substitui a variável **FirstWeekDay**. Para indicar a semana que começa em outro dia, especifique um sinalizador entre 0 e 6.

Valores de first_week_day

Dia	Valor
Segunda-feira	0
Terça-feira	1
Quarta-feira	2
Quinta-feira	3
Sexta-feira	4
Sábado	5
Domingo	6

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplos de funções

Exemplo	Resultado
`inweek ('01/12/2006', '01/14/2006', 0)`	Retorna TRUE
`inweek ('01/12/2006', '01/20/2006', 0 )`	Retorna FALSE
`inweek ('01/12/2006', '01/14/2006', -1 )`	Retorna FALSE
`inweek ('01/07/2006', '01/14/2006', -1)`	Retorna TRUE

Exemplo	Resultado
`inweek ('01/12/2006', '01/09/2006', 0, 3)`	Retorna FALSE porque `first_week_day` está especificado como 3 (quinta-feira), o que torna 01/12/2006 o primeiro dia da semana seguinte à semana contendo 01/09/2006.

Estes tópicos podem ajudar você a trabalhar com essa função:

Tópicos relacionados

Tópico	Sinalizador/valor padrão	Descrição
FirstWeekDay (page 226)	6 / Domingo	Define o dia de início de cada semana.

Exemplo 1: Nenhum argumento adicional

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações para o mês de janeiro de 2022, que é carregado em uma tabela denominada "`Transactions`".
- A variável de sistema `FirstWeekDay` que é definida como `6` (domingo).
- Um carregamento anterior que contém o seguinte:
  - A função `inweek()`, definida como o campo "`in_week`", que determina quais transações ocorreram na semana de 14 de janeiro de 2022.
  - A função `weekday()`, definida como o campo "`week_day`", que mostra qual dia da semana corresponde a cada data.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `week_day`
- `in_week`

Tabela de resultados

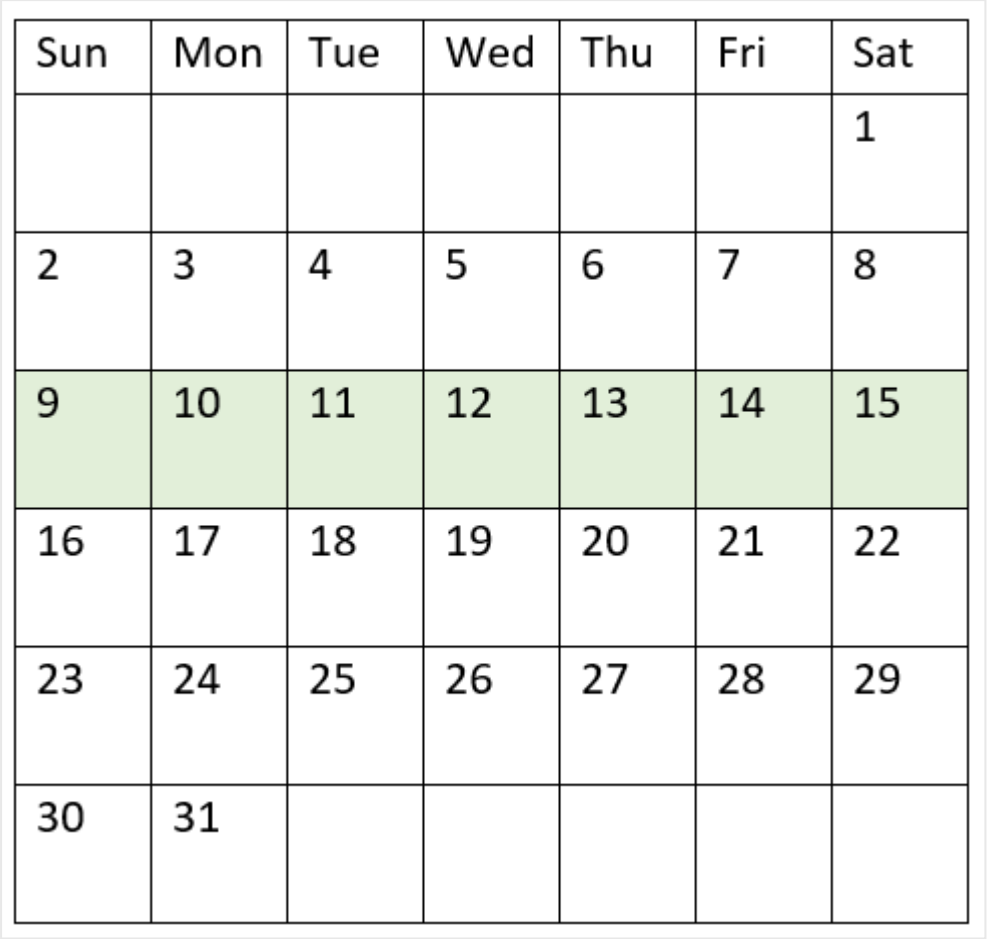
date	week_day	in_week
01/02/2022	Dom	0
01/05/2022	Qua	0
01/06/2022	Qui	0
01/08/2022	Sáb	0
01/09/2022	Dom	-1
01/10/2022	Mon	-1
01/11/2022	Ter	-1
01/12/2022	Qua	-1
01/13/2022	Qui	-1
01/14/2022	Fri	-1
01/15/2022	Sáb	-1
01/16/2022	Dom	0

date	week_day	in_week
01/17/2022	Mon	0
01/18/2022	Ter	0
01/26/2022	Qua	0
01/27/2022	Qui	0
01/28/2022	Fri	0
01/29/2022	Sáb	0
01/30/2022	Dom	0
01/31/2022	Mon	0

O campo "in_week" é criado na instrução de carregamento anterior usando a função `inweek()`. O primeiro argumento identifica qual campo está sendo avaliado. O segundo argumento é uma data codificada para 14 de janeiro, que é a `base_date`. O argumento `base_date` funciona com a variável de sistema `FirstWeekDay` para identificar a semana comparadora. `period_no` de `0`, significando que a função não está comparando semanas anteriores ou posteriores à semana segmentada, é o argumento final.

A variável de sistema `FirstWeekDay` determina que as semanas começam em um domingo e terminam em um sábado. Portanto, janeiro seria dividido em semanas de acordo com o diagrama abaixo, com as datas entre 9 e 15 de janeiro fornecendo o período válido para o cálculo de `inweek()`:

Diagrama do calendário com o intervalo da função `inweek()` destacado



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					

Qualquer transação que ocorra entre 9 e 15 de janeiro retorna um resultado booleano de TRUE.

Exemplo 2: period_no

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- O mesmo conjunto de dados contendo um conjunto de transações para 2022 é carregado em uma tabela denominada "`Transactions`".
- A variável de sistema `FirstWeekDay` que é definida como 6 (domingo).
- Um carregamento anterior que contém o seguinte:
    - A função `inweek ()`, definida como o campo "`prev_week`", que determina quais transações ocorreram uma semana inteira antes da semana de 14 de janeiro de 2022.

- A função weekday(), definida como o campo "week_day", que mostra qual dia da semana corresponde a cada data.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

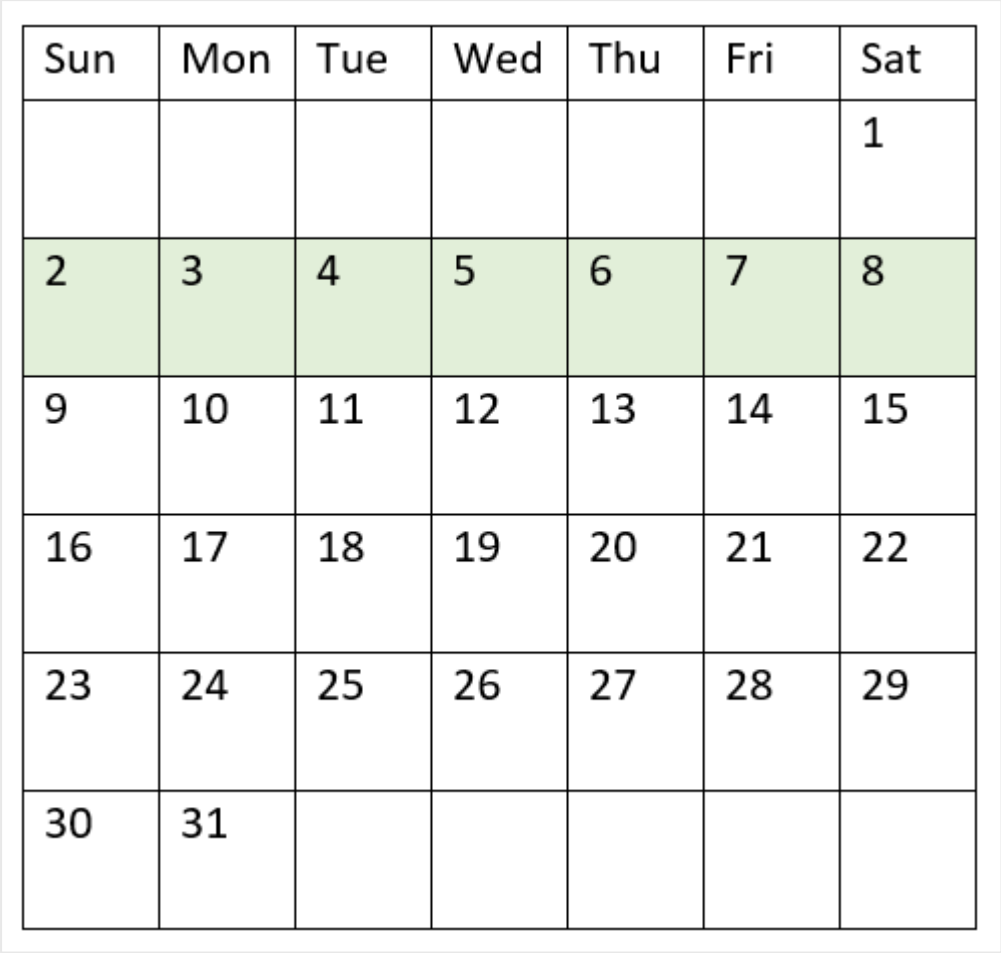
- date
- week_day
- prev_week

Tabela de resultados

date	week_day	prev_week
01/02/2022	Dom	-1
01/05/2022	Qua	-1
01/06/2022	Qui	-1
01/08/2022	Sáb	-1
01/09/2022	Dom	0
01/10/2022	Mon	0
01/11/2022	Ter	0
01/12/2022	Qua	0
01/13/2022	Qui	0
01/14/2022	Fri	0
01/15/2022	Sáb	0
01/16/2022	Dom	0
01/17/2022	Mon	0
01/18/2022	Ter	0
01/26/2022	Qua	0
01/27/2022	Qui	0
01/28/2022	Fri	0
01/29/2022	Sáb	0
01/30/2022	Dom	0
01/31/2022	Mon	0

Usar `-1` como argumento `period_no` na função `inweek()` altera os limites da semana comparadora em sete dias completos. Com um `period_no` de `0`, a semana seria entre 9 e 15 de janeiro. Mas, neste exemplo, o `period_no` de `-1` desloca o limite inicial e final desse segmento para trás em uma semana. Os limites da data passam de 2 de janeiro a 8 de janeiro.

Diagrama do calendário com o intervalo da função `inweek()` destacado



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					

Portanto, qualquer transação que ocorra entre 2 de janeiro e 8 de janeiro retornará um resultado booleano de TRUE.

Exemplo 3: first_week_day

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- O mesmo conjunto de dados contendo um conjunto de transações para 2022 é carregado em uma tabela denominada "`Transactions`".
- A variável de sistema `FirstWeekDay` que é definida como 6 (domingo).
- Um carregamento anterior que contém o seguinte:
    - A função `inweek()`, definida como o campo "`in_week`", que determina quais transações ocorreram na semana de 14 de janeiro de 2022.

- A função weekday(), definida como o campo "week_day", que mostra qual dia da semana corresponde a cada data.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- date
- week_day
- in_week

Tabela de resultados

date	week_day	in_week
01/02/2022	Dom	0
01/05/2022	Qua	0
01/06/2022	Qui	0
01/08/2022	Sáb	0
01/09/2022	Dom	0
01/10/2022	Mon	-1
01/11/2022	Ter	-1
01/12/2022	Qua	-1
01/13/2022	Qui	-1
01/14/2022	Fri	-1
01/15/2022	Sáb	-1
01/16/2022	Dom	-1
01/17/2022	Mon	0
01/18/2022	Ter	0
01/26/2022	Qua	0
01/27/2022	Qui	0
01/28/2022	Fri	0
01/29/2022	Sáb	0
01/30/2022	Dom	0
01/31/2022	Mon	0

O uso de `0` como argumento `first_week_day` na função `inweek()` substitui a variável de sistema `FirstWeekDay` e define segunda-feira como o primeiro dia da semana.

Diagrama do calendário com o intervalo da função `inweek()` destacado

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					

Portanto, qualquer transação que ocorra entre 10 e 16 de janeiro retornará um resultado booleano de TRUE.

Exemplo 4: Exemplo de objeto de gráfico

Script de carregamento e expressão de gráfico

Visão geral

São usados o mesmo conjunto de dados e cenário do primeiro exemplo.

No entanto, neste exemplo, o conjunto de dados permanece inalterado e é carregado no aplicativo. Crie uma medida na tabela de resultados para determinar quais transações ocorreram na semana de 14 de janeiro de 2022.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão:

- `date`

Crie as seguintes medidas:

- `=inweek (date,'01/14/2022',0)`, para calcular se as transações ocorreram na mesma semana de 14 de janeiro.
- `=weekday(date)`, para mostrar qual dia da semana corresponde a cada data.

Tabela de resultados

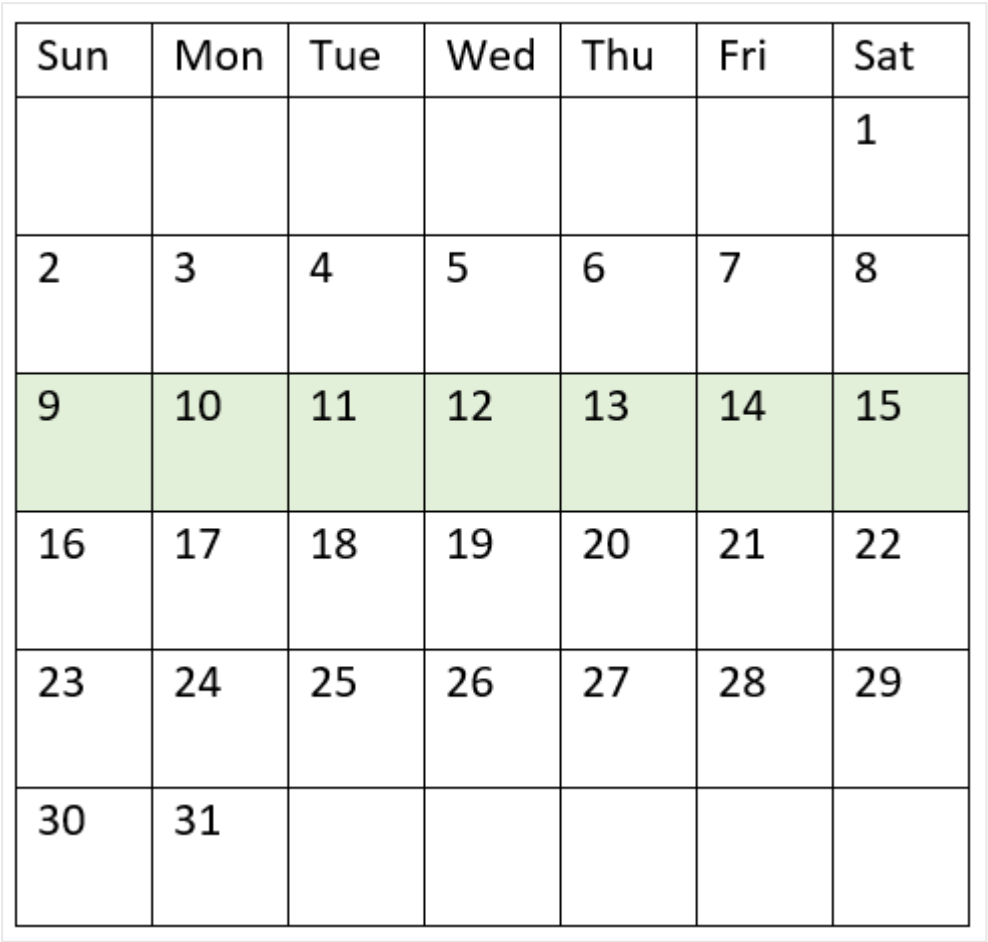
date	week_day	=inweek (date,'01/14/2022',0)
01/02/2022	Dom	0
01/05/2022	Qua	0
01/06/2022	Qui	0
01/08/2022	Sáb	0
01/09/2022	Dom	-1
01/10/2022	Mon	-1
01/11/2022	Ter	-1

date	week_day	=inweek (date,'01/14/2022',0)
01/12/2022	Qua	-1
01/13/2022	Qui	-1
01/14/2022	Fri	-1
01/15/2022	Sáb	-1
01/16/2022	Dom	0
01/17/2022	Mon	0
01/18/2022	Ter	0
01/26/2022	Qua	0
01/27/2022	Qui	0
01/28/2022	Fri	0
01/29/2022	Sáb	0
01/30/2022	Dom	0
01/31/2022	Mon	0

A medida "in_week" é criada no gráfico usando a função `inweek()`. O primeiro argumento identifica qual campo está sendo avaliado. O segundo argumento é uma data codificada para 14 de janeiro, que é a `base_date`. O argumento `base_date` funciona com a variável de sistema `FirstWeekDay` para identificar a semana comparadora. Um `period_no` de `0` é o argumento final.

A variável de sistema `FirstWeekDay` determina que as semanas começam em um domingo e terminam em um sábado. Portanto, janeiro seria dividido em semanas de acordo com o diagrama abaixo, com as datas entre 9 e 15 de janeiro fornecendo o período válido para o cálculo de `inweek()`:

Diagrama do calendário com o intervalo da função `inweek()` destacado



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					

Qualquer transação que ocorra entre 9 e 15 de janeiro retorna um resultado booleano de TRUE.

Exemplo 5: Cenário

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados que é carregado em uma tabela denominada "`Products`".
- A tabela contém os seguintes campos:
    - product ID
    - product type
    - manufacture date
    - cost price

Foi identificado que, devido a um erro de equipamento, os produtos fabricados na semana de 12 de janeiro estavam com defeito. O usuário final gostaria de um gráfico que exibisse, por semana, o status de quais produtos fabricados estavam “com defeito” ou “sem defeito”, bem como o custo dos produtos fabricados naquela semana.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão:

- `=weekname(manufacture_date)`

Crie as seguintes medidas:

- `=if(only(inweek(manufacture_date,makedate(2022,01,12),0)),'Defective','Faultless')`, para identificar quais produtos estão com defeito e quais estão sem defeito usando a função `inweek()`.
- `=sum(cost_price)`, para mostrar a soma do custo de cada produto.

Faça o seguinte:

1. Defina o **Formato numérico** da medida como **Dinheiro**.
2. Em **Aparência**, desative **Totais**.

Tabela de resultados

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

A função `inweek()` retorna um valor booleano ao avaliar as datas de fabricação de cada um dos produtos. Para qualquer produto fabricado na semana de 12 de janeiro, a função `inweek()` retorna um valor booleano de TRUE e marca os produtos como “Defective”. Para qualquer produto que retorne um valor de FALSE e, portanto, não fabricado naquela semana, ela marca os produtos como “Faultless”.

inweektodate

Essa função retornará True se **timestamp** estiver na parte da semana que contém a **base_date** até e inclusive o último milissegundo da **base_date**.

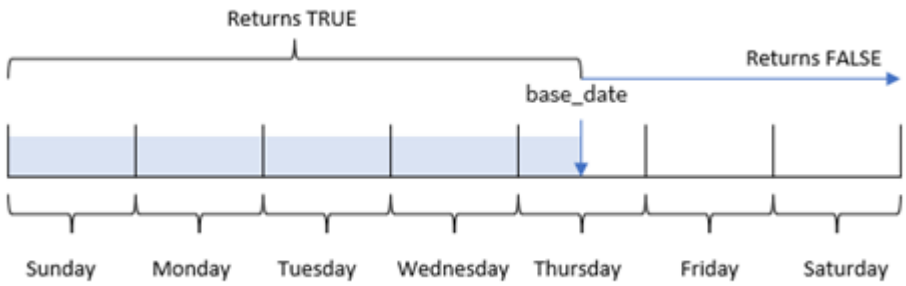
Sintaxe:

```
InWeekToDate (timestamp, base_date, period_no [, first_week_day])
```

Tipo de dados de retorno: Booleano

> *No Qlik Sense, o valor booleano “true” é representado por -1, e o valor falso é representado por 0.*

Diagrama da função inweektodate



A função `inweektodate()` usa o parâmetro `base_date` para identificar uma data limite máxima de um segmento de semana, bem como sua data correspondente para o início da semana, que se baseia na `variável` do sistema FirstWeekDay (ou no parâmetro `first_week_day` definido pelo usuário). Assim que o segmento dessa semana tiver sido definido, a função retornará resultados booleanos ao comparar os valores de data prescritos com esse segmento.

Quando usar

A função `inweektodate()` retorna um resultado booleano. Normalmente, esse tipo de função será usado como uma condição em uma expressão `if`. Isso retornará uma agregação ou cálculo, dependendo de se uma data avaliada ocorreu durante a semana em questão, até e incluindo uma data específica.

Por exemplo, a função `inweektodate()` pode ser usada para calcular todas as vendas feitas durante uma semana especificada até uma determinada data.

Argumentos

Argumento	Descrição
timestamp	A data que você deseja comparar com **base_date**.
base_date	Data que é usada para avaliar a semana.
period_no	A semana pode ser deslocada por **period_no**. **period_no** é um inteiro, em que o valor 0 indica a semana que contém **base_date**. Os valores negativos em **period_no** indicam semanas precedentes e os valores positivos indicam semanas subsequentes.
first_week_day	Por padrão, o primeiro dia da semana é domingo (conforme determinado pela variável de sistema FirstWeekDay), começando à meia-noite entre sábado e domingo. O parâmetro **first_week_day** substitui a variável **FirstWeekDay**. Para indicar a semana que começa em outro dia, especifique um sinalizador entre 0 e 6. Para uma semana começando na segunda-feira e terminando no domingo, use um sinalizador de 0 para segunda-feira, 1 para terça-feira, 2 para quarta-feira, 3 para quinta-feira, 4 para sexta-feira, 5 para sábado e 6 para domingo.

Exemplos de funções

Exemplo	Interação
`inweektodate ('01/12/2006', '01/12/2006', 0)`	Returna `TRUE`.
`inweektodate ('01/12/2006', '01/11/2006', 0)`	Retorna `FALSE`.
`inweektodate ('01/12/2006', '01/18/2006', -1)`	Returna `FALSE`. Como `period_no` é especificado como -1, os dados efetivos com base nos quais `timestamp` é medido são 01/11/2006.
`inweektodate ('01/11/2006', '01/12/2006', 0, 3 )`	Retorna `FALSE`, já que `first_week_day` está especificado como `3` (quinta-feira), o que torna 01/12/2006 o primeiro dia da semana seguinte à semana que contém 01/12/2006.

Estes tópicos podem ajudar você a trabalhar com essa função:

Tópicos relacionados

Tópico	Sinalizador/valor padrão	Descrição
FirstWeekDay (page 226)	6 / Domingo	Define o dia de início de cada semana.

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplo 1: Sem argumentos adicionais

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações para o mês de janeiro de 2022, que é carregado em uma tabela denominada "`Transactions`".
- O campo de dados fornecido no formato `TimestampFormat='M/D/YYYY h:mm:ss[.fff]'`.
- A criação de um campo, `in_week_to_date`, que determina quais transações ocorreram na semana até 14 de janeiro de 2022.
- A criação de um campo adicional, chamado `weekday`, usando a função `weekday()`. Esse novo campo é criado para mostrar qual dia da semana corresponde a cada data.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `week_day`
- `in_week_to_date`

Tabela de resultados

date	week_day	in_week_to_date
2022-01-02 12:22:06	Dom	0
2022-01-05 01:02:30	Qua	0
2022-01-06 15:36:20	Qui	0
2022-01-08 10:58:35	Sáb	0
2022-01-09 08:53:32	Dom	-1
2022-01-10 21:13:01	Seg	-1
2022-01-11 00:57:13	Ter	-1
2022-01-12 09:26:02	Qua	-1

date	week_day	in_week_to_date
2022-01-13 15:05:09	Qui	-1
2022-01-14 18:44:57	Sex	-1
2022-01-15 06:10:46	Sáb	0
2022-01-16 06:39:27	Dom	0
2022-01-17 10:44:16	Seg	0
2022-01-18 18:48:17	Ter	0
2022-01-26 04:36:03	Qua	0
2022-01-27 08:07:49	Qui	0
2022-01-28 12:24:29	Sex	0
2022-01-30 11:56:56	Dom	0
2022-01-30 14:40:19	Dom	0
2022-01-31 05:28:21	Seg	0

O campo `in_week_to_date` é criado na instrução de carregamento anterior usando a função `inweektodate()`. O primeiro argumento fornecido identifica qual campo está sendo avaliado. O segundo argumento é uma data codificada para 14 de janeiro, que é a `base_date` que identifica qual semana segmentar e que define o limite final desse segmento. Um `period_no` de 0 é o argumento final, o que significa que a função não está comparando semanas anteriores ou posteriores à semana segmentada.

A variável de sistema `FirstWeekDay` determina que as semanas começam em um domingo e terminam em um sábado. Portanto, janeiro seria dividido em semanas de acordo com o diagrama abaixo, com as datas entre 9 e 14 de janeiro fornecendo o período válido para o cálculo de `inweekdodate()`:

Diagrama de calendário mostrando datas de transação que retornariam um resultado booleano de TRUE

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					

Qualquer transação que ocorra entre 9 e 14 de janeiro retorna um resultado booleano de TRUE. As transações antes e depois das datas retornam um resultado booleano de FALSE.

Exemplo 2: period_no

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- O mesmo conjunto de dados e cenário do primeiro exemplo.
- A criação de um campo, `prev_week_to_date`, que determina quais transações ocorreram uma semana inteira antes do término do segmento de semana em 14 de janeiro de 2022.
- A criação de um campo adicional, chamado `weekday`, usando a função `weekday()`. Ele é criado para mostrar qual dia da semana corresponde a cada data.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `week_day`
- `prev_week_to_date`

Tabela de resultados

date	week_day	prev_week_to_date
2022-01-02 12:22:06	Dom	-1
2022-01-05 01:02:30	Qua	-1
2022-01-06 15:36:20	Qui	-1
2022-01-08 10:58:35	Sáb	0
2022-01-09 08:53:32	Dom	0
2022-01-10 21:13:01	Seg	0
2022-01-11 00:57:13	Ter	0
2022-01-12 09:26:02	Qua	0
2022-01-13 15:05:09	Qui	0
2022-01-14 18:44:57	Sex	0
2022-01-15 06:10:46	Sáb	0
2022-01-16 06:39:27	Dom	0
2022-01-17 10:44:16	Seg	0
2022-01-18 18:48:17	Ter	0
2022-01-26 04:36:03	Qua	0
2022-01-27 08:07:49	Qui	0
2022-01-28 12:24:29	Sex	0
2022-01-30 11:56:56	Dom	0
2022-01-30 14:40:19	Dom	0
2022-01-31 05:28:21	Seg	0

Um valor de `period_no` -1 indica que a função `inweektodate ()` compara o segmento de trimestre de entrada com a semana anterior. O segmento de semana equivale inicialmente a um período entre 9 e 14 de janeiro. `period_no` então desloca o limite inicial e final desse segmento para uma semana antes, fazendo com que os limites de data se tornem de 2 a 7 de janeiro.

Diagrama de calendário mostrando datas de transação que retornariam um resultado booleano de TRUE

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					

Portanto, qualquer transação que ocorra entre 2 e 8 de janeiro (sem incluir 8 de janeiro) retornará um resultado booleano de TRUE.

Exemplo 3: first_week_day

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- O mesmo conjunto de dados e cenário do primeiro exemplo.
- A criação de um campo, `in_week_to_date`, que determina quais transações ocorreram na semana até 14 de janeiro de 2022.
- A criação de um campo adicional, chamado `weekday`, usando a função `weekday()`. Ele é criado para mostrar qual dia da semana corresponde a cada data.

Neste exemplo, consideramos segunda-feira o primeiro dia da semana.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `week_day`
- `in_week_to_date`

Tabela de resultados

date	week_day	in_week_to_date
2022-01-02 12:22:06	Dom	0
2022-01-05 01:02:30	Qua	0
2022-01-06 15:36:20	Qui	0
2022-01-08 10:58:35	Sáb	0
2022-01-09 08:53:32	Dom	0
2022-01-10 21:13:01	Seg	-1
2022-01-11 00:57:13	Ter	-1
2022-01-12 09:26:02	Qua	-1
2022-01-13 15:05:09	Qui	-1

date	week_day	in_week_to_date
2022-01-14 18:44:57	Sex	-1
2022-01-15 06:10:46	Sáb	0
2022-01-16 06:39:27	Dom	0
2022-01-17 10:44:16	Seg	0
2022-01-18 18:48:17	Ter	0
2022-01-26 04:36:03	Qua	0
2022-01-27 08:07:49	Qui	0
2022-01-28 12:24:29	Sex	0
2022-01-30 11:56:56	Dom	0
2022-01-30 14:40:19	Dom	0
2022-01-31 05:28:21	Seg	0

Usando 0 como argumento `first_week_day` na função `inweektodate()`, o argumento da função substitui a variável do sistema `FirstWeekDay` e define segunda-feira como o primeiro dia da semana.

Diagrama de calendário mostrando datas de transação que retornariam um resultado booleano de TRUE

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						

Portanto, qualquer transação que ocorra entre 10 e 14 de janeiro retornará um resultado booleano de TRUE, enquanto as transações com datas fora desses limites retornarão um valor de FALSE.

Exemplo 4: Exemplo de objeto de gráfico

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém o mesmo conjunto de dados e cenário do primeiro exemplo. No entanto, neste exemplo, o conjunto de dados inalterado é carregado no aplicativo. O cálculo que determina quais transações ocorreram na semana até 14 de janeiro de 2022 é criado como uma medida no objeto de gráfico.

Script de carregamento

```
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
];
```

Resultados

Faça o seguinte:

1. Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão: `date`.

2. Para calcular se as transações ocorreram na mesma semana até 14 de janeiro, crie a seguinte medida: `=inweektodate(date,'01/14/2022',0)`
3. Para mostrar qual dia da semana corresponde a cada data, crie uma medida adicional: `=weekday(date)`

Tabela de resultados

date	week_day	in_week_to_date
2022-01-02 12:22:06	Dom	0
2022-01-05 01:02:30	Qua	0
2022-01-06 15:36:20	Qui	0
2022-01-08 10:58:35	Sáb	0
2022-01-09 08:53:32	Dom	-1
2022-01-10 21:13:01	Seg	-1
2022-01-11 00:57:13	Ter	-1
2022-01-12 09:26:02	Qua	-1
2022-01-13 15:05:09	Qui	-1
2022-01-14 18:44:57	Sex	-1
2022-01-15 06:10:46	Sáb	0
2022-01-16 06:39:27	Dom	0
2022-01-17 10:44:16	Seg	0
2022-01-18 18:48:17	Ter	0
2022-01-26 04:36:03	Qua	0
2022-01-27 08:07:49	Qui	0
2022-01-28 12:24:29	Sex	0
2022-01-30 11:56:56	Dom	0
2022-01-30 14:40:19	Dom	0
2022-01-31 05:28:21	Seg	0

O campo `in_week_to_date` é criado como uma medida no objeto de gráfico usando a função `inweektodate()`. O primeiro argumento fornecido identifica qual campo está sendo avaliado. O segundo argumento é uma data codificada para 14 de janeiro, que é a `base_date` que identifica qual semana segmentar e que define o limite final desse segmento. Um `period_no` de 0 é o argumento final, o que significa que a função não está comparando semanas anteriores ou posteriores à semana segmentada.

A variável de sistema `FirstWeekDay` determina que as semanas começam em um domingo e terminam em um sábado. Portanto, janeiro seria dividido em semanas de acordo com o diagrama abaixo, com as datas entre 9 e 14 de janeiro fornecendo o período válido para o cálculo de `inweekdodate()`:

Diagrama de calendário mostrando datas de transação que retornariam um resultado booleano de TRUE

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					

Qualquer transação que ocorra entre 9 e 14 de janeiro retorna um resultado booleano de TRUE. As transações antes e depois das datas retornam um resultado booleano de FALSE.

Exemplo 5: Cenário

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados que é carregado em uma tabela denominada `Products`.
- Informações sobre o ID do produto, data de fabricação e preço de custo.

Foi identificado que, devido a um erro de equipamento, os produtos fabricados na semana de 12 de janeiro estavam com defeito. O problema foi resolvido em 13 de janeiro. O usuário final deseja um objeto de gráfico que mostre, por semana, o status que indica se os produtos fabricados estão "com defeito" ou "sem defeito", bem como o custo dos produtos fabricados naquela semana.

Script de carregamento

```
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
];
```

Resultados

Faça o seguinte:

1. Carregue os dados e abra uma pasta. Crie uma nova tabela. Crie uma dimensão para mostrar os nomes das semanas:
   `=weekname(manufacture_date)`
2. Em seguida, crie uma dimensão para identificar quais dos produtos estão com defeito e quais estão sem defeito:
   `=if(inWeektodate(manufacture_date,makedate(2022,01,12),0),'Defective','Faultless')`
3. Crie uma medida para somar o `cost_price` dos produtos:
   `=sum(cost_price)`
4. Defina o **Formato numérico** da medida como **Dinheiro**.

Tabela de resultados

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

A função `inweektodate()` retorna um valor booleano ao avaliar as datas de fabricação de cada um dos produtos. Para os casos que retornam um valor booleano de `TRUE`, ela marca os produtos como `'Defective'`. Para qualquer produto que retorne um valor de `FALSE` e, portanto, não tenha sido fabricado na semana até 12 de janeiro, ela marca os produtos como `'Faultless'`.

inyear

Essa função retornará True se **timestamp** estiver dentro do ano que contém a **base_date**.

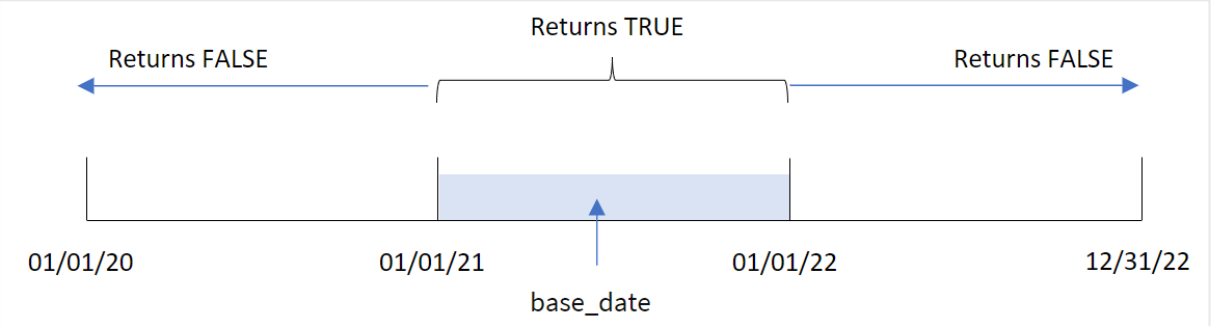
Sintaxe:

```
InYear (timestamp, base_date, period_no [, first_month_of_year])
```

Tipo de dados de retorno: Booleano

No Qlik Sense, o valor booleano “true” é representado por -1, e o valor falso é representado por 0.

Diagrama do intervalo da função `inyear()`



A função `inyear()` retorna um resultado booleano ao comparar os valores de data selecionados com um ano definido pela `base_date`.

Quando usar

A função `inyear()` retorna um resultado booleano. Normalmente, esse tipo de função será usado como uma condição em um `if expression`. Isso retorna uma agregação ou cálculo dependendo se uma data avaliada ocorreu no ano em questão. Por exemplo, a função `inyear()` pode ser usada para identificar todas as vendas que ocorreram em um ano definido.

Argumentos

Argumento	Descrição
timestamp	A data que você deseja comparar com **base_date**.
base_date	Data que é usada para avaliar o ano.
period_no	O ano pode ser deslocado por **period_no**. **period_no** é um inteiro, em que o valor 0 indica o ano que contém **base_date**. Valores negativos em **period_no** indicam anos precedentes e valores positivos indicam anos sucessivos.
first_month_of_year	Se desejar trabalhar com anos (fiscais) que não comecem em janeiro, indique um valor entre 2 e 12 em **first_month_of_year**.

Você pode usar os seguintes valores para definir o primeiro mês do ano no argumento `first_month_of_year`:

Valores first_month_of_year

Month	Valor
Fevereiro	2
Março	3
Abril	4
Maio	5
Junho	6
Julho	7
Agosto	8
Setembro	9
Outubro	10
Novembro	11
Dezembro	12

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplos de funções

Exemplo	Resultado
`inyear ('01/25/2013', '01/01/2013', 0 )`	Retorna TRUE
`inyear ('01/25/2012', '01/01/2013', 0)`	Retorna FALSE
`inyear ('01/25/2013', '01/01/2013', -1)`	Retorna FALSE

Exemplo	Resultado
`inyear ('01/25/2012', '01/01/2013', -1 )`	Retorna TRUE
`inyear ('01/25/2013', '01/01/2013', 0, 3)`	Retorna TRUE Os valores de `base_date` e `first_month_of_year` especificam que esse carimbo de data/hora deve estar entre 03/01/2012 e 28/02/2013
`inyear ('03/25/2013', '07/01/2013', 0, 3 )`	Retorna TRUE

Exemplo 1: Exemplo básico

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações entre 2020 e 2022, que é carregado em uma tabela denominada "`Transactions`".
- Um carregamento anterior que contém a função `inyear()` definida como o `campo` "in_year" e determina quais transações ocorreram no mesmo ano de 26 de julho de 2021.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

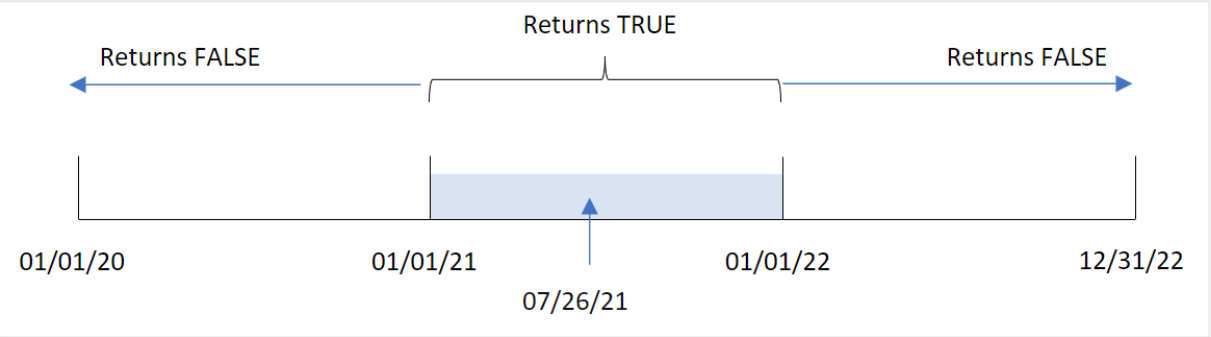
- `date`
- `in_year`

Tabela de resultados

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

O campo "in_year" é criado na instrução de carregamento anterior usando a função inyear(). O primeiro argumento identifica qual campo está sendo avaliado. O segundo argumento é uma data codificada para 26 de julho de 2021, que é a base_date que determina o ano comparador. Um period_no de 0 é o argumento final, significando que a função inyear() não compara os anos anteriores ou posteriores ao ano.

Diagrama do intervalo da função inyear()com 26 de julho como data base



Qualquer transação que ocorra em 2021 retorna um resultado booleano de TRUE.

Exemplo 2: period_no

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações entre 2020 e 2022, que é carregado em uma tabela denominada "Transactions".
- Um carregamento anterior que contém a função inyear() definida como o campo "previous_year" e determina quais transações ocorreram no ano anterior ao ano que contém 26 de julho de 2021.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `previous_year`

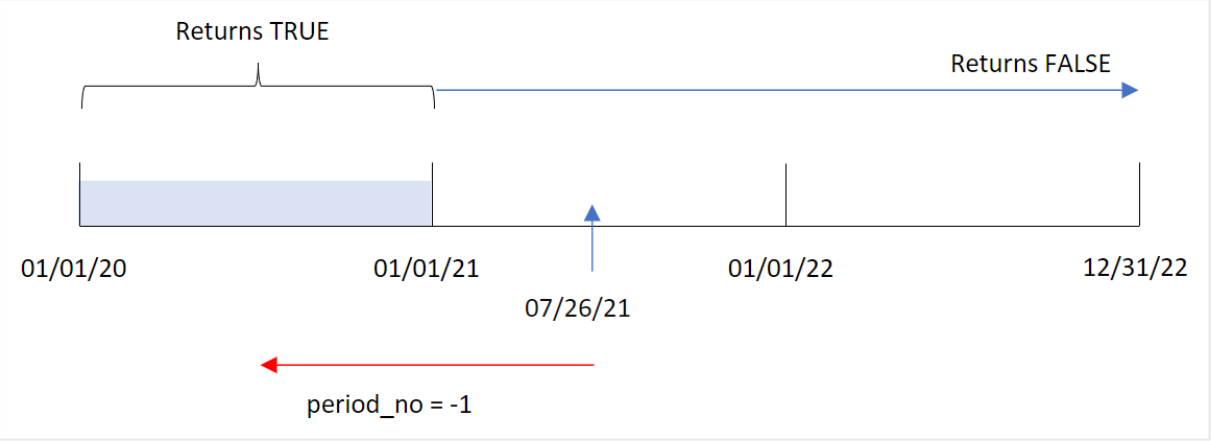
Tabela de resultados

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

date	previous_year
07/26/2021	0
12/27/2021	0
06/06/2022	0
07/18/2022	0
11/14/2022	0
12/12/2022	0

Usar -1 como argumento period_no na função inyear() altera os limites do ano comparador em um ano inteiro. 2021 é inicialmente identificado como o ano comparador. period_no desloca o ano comparador em um, tornando 2020 o ano comparador.

Diagrama do intervalo da função inyear() com o argumento period_no definido como -1



Portanto, qualquer transação que ocorra em 2020 retorna um resultado booleano de TRUE.

Exemplo 3: first_month_of_year

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações entre 2020 e 2022, que é carregado em uma tabela denominada "Transactions".
- Um carregamento anterior que contém a função inyear() definida como o campo "in_year" e determina quais transações ocorreram no mesmo ano de 26 de julho de 2021.

No entanto, neste exemplo, a política organizacional é que março seja o primeiro mês do exercício financeiro.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `in_year`

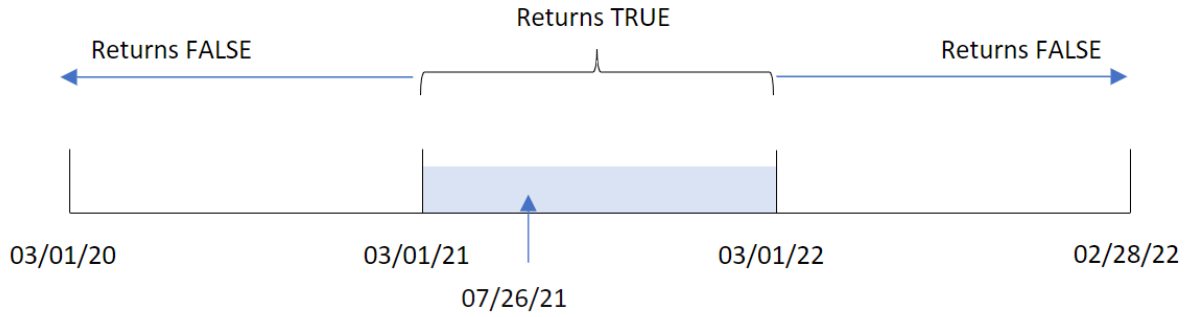
Tabela de resultados

date	in_year
01/13/2020	0
02/26/2020	0
03/27/2020	0
04/16/2020	0

date	in_year
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	-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

Usar 3 como argumento `first_month_of_year` na função `inyear()` começa o ano em 1º de março e termina no final de fevereiro.

Diagrama do intervalo da função `inyear()` com o mês de março definido como o primeiro mês do ano



Portanto, qualquer transação que ocorra entre 1º de março de 2021 e 1º de março de 2022 retornará um resultado booleano de TRUE.

Exemplo 4: Exemplo de objeto de gráfico

Script de carregamento e expressão de gráfico

Visão geral

São usados o mesmo conjunto de dados e cenário do primeiro exemplo.

No entanto, neste exemplo, o conjunto de dados permanece inalterado e é carregado no aplicativo. O cálculo que determina se as transações ocorreram no mesmo ano de 26 de julho de 2021 é criado como uma medida em um objeto de gráfico do aplicativo.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão:

- `date`

Para calcular se as transações ocorreram no mesmo ano de 26 de julho de 2021, crie a seguinte medida:

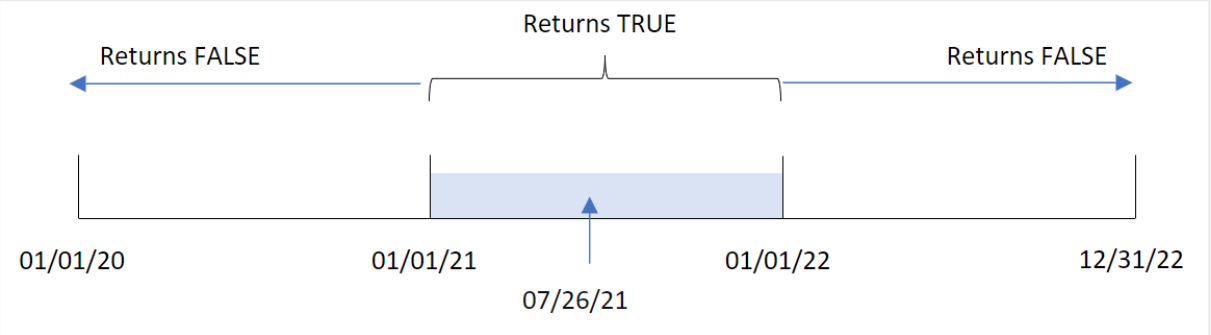
- `=inyear(date,'07/26/2021', 0)`

Tabela de resultados

date	=inyear(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
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

O campo “in_year” é criado no gráfico usando a função `inyear()`. O primeiro argumento identifica qual campo está sendo avaliado. O segundo argumento é uma data codificada para 26 de julho de 2021, que é a `base_date` que determina o ano comparador. Um `period_no` de `0` é o argumento final, significando que a função `inyear()` não compara os anos anteriores ou posteriores ao ano.

Diagrama do intervalo da função `inyear()`com 27 de julho como data base



Qualquer transação que ocorra em 2021 retorna um resultado booleano de TRUE.

Exemplo 5: Cenário

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados que é carregado em uma tabela denominada "`Products`".
- A tabela contém os seguintes campos:
    - product ID
    - product type
    - manufacture date
    - cost price

O usuário final gostaria de um objeto de gráfico que exiba, por tipo de produto, o custo dos produtos fabricados em 2021.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão:

- `product_type`

Para calcular a soma de cada produto que foi fabricado em 2021, crie a seguinte medida:

- `=sum(if(InYear(manufacture_date,makedate(2021,01,01),0),cost_price,0))`

Faça o seguinte:

1. Defina o **Formato numérico** da medida como **Dinheiro**.
2. Em **Aparência**, desative **Totais**.

Tabela de resultados

product_type	=sum(if(InYear(manufacture_date,makedate(2021,01,01),0),cost_price,0))
produto A	$95.93
produto B	$128.66
produto C	$61.89
produto D	$171.21

A função `inyear()` retorna um valor booleano ao avaliar as datas de fabricação de cada um dos produtos. Para qualquer produto fabricado em 2021, a função `inyear()` retorna um valor booleano de TRUE e mostra a soma de `cost_price`.

inyeartodate

Esta função retornará True se **timestamp** estiver na parte da parte do ano que contém **base_date** até e inclusive o último milissegundo de **base_date**.

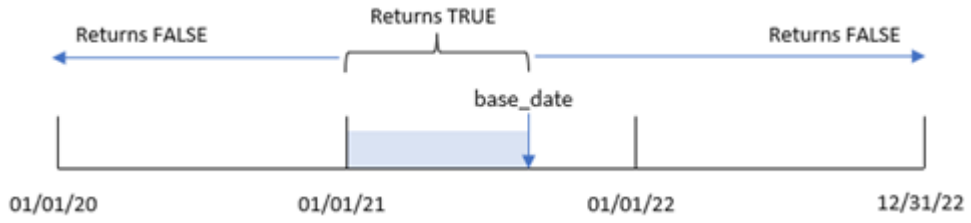
Sintaxe:

```
InYearToDate (timestamp, base_date, period_no[, first_month_of_year])
```

Tipo de dados de retorno: Booleano

No Qlik Sense, o valor booleano "true" é representado por -1, e o valor falso é representado por 0.

Diagrama da função inyeartodate



A função `inyeartodate()` segmentará uma parte específica do ano com a `base_date`, identificando a data máxima permitida para esse segmento de ano. Em seguida, a função avalia se um campo ou valor de data se enquadra nesse segmento e retorna um resultado booleano.

Argumentos

Argumento	Descrição
timestamp	A data que você deseja comparar com **base_date**.
base_date	Data que é usada para avaliar o ano.
period_no	O ano pode ser deslocado por **period_no**. **period_no** é um inteiro, em que o valor 0 indica o ano que contém **base_date**. Valores negativos em **period_no** indicam anos precedentes e valores positivos indicam anos sucessivos.
first_month_of_year	Se desejar trabalhar com anos (fiscais) que não comecem em janeiro, indique um valor entre 2 e 12 em **first_month_of_year**.

Quando usar

A função `inyeartodate()` retorna um resultado booleano. Normalmente, esse tipo de função será usado como uma condição em uma expressão `if`. Isso retornaria uma agregação ou um cálculo dependendo de uma data avaliada ter ocorrido no ano até e incluindo a data em questão.

Por exemplo, a função `inyeartodate()` pode ser usada para identificar todos os equipamentos fabricados em um ano até uma data específica.

Esses exemplos usam o formato de data MM/DD/AAAA. O formato de data é especificado no comando `SET DateFormat` na parte superior do seu script de carregamento de dados. Altere o formato nos exemplos para atender às suas necessidades.

Exemplos de funções

Exemplo	Resultado
`inyeartodate ('01/25/2013', '02/01/2013', 0)`	Retorna TRUE.
`inyeartodate ('01/25/2012', '01/01/2013', 0)`	Retorna FALSE.
`inyeartodate ('01/25/2012', '02/01/2013', -1)`	Retorna TRUE.
`inyeartodate ('11/25/2012', '01/31/2013', 0, 4)`	Returna TRUE. O valor de `timestamp` cai no ano fiscal que começa no quarto mês e antes do valor de `base_date`.
`inyeartodate ('3/31/2013', '01/31/2013', 0, 4 )`	Returna FALSE. Comparado com o exemplo anterior, o valor de `timestamp` ainda está dentro do ano fiscal, mas é depois que o valor de `base_date`, assim, fica fora de parte do ano.

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplo 1: Sem argumentos adicionais

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações entre 2020 e 2022, que é carregado em uma tabela denominada `Transactions`.
- O campo de data fornecido no formato da `variável` de sistema DateFormat (MM/DD/AAAA).
- A criação de um campo, `in_year_to_date`, que determina quais transações ocorreram no ano até 26 de julho de 2021.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `in_year_to_date`

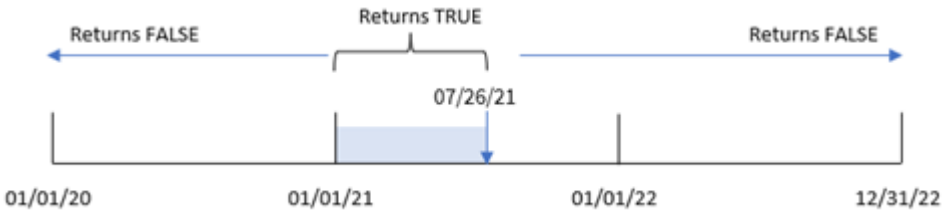
Tabela de resultados

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	-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

O campo `in_year_to_date` é criado na instrução de carregamento anterior usando a função `inyeartodate()`. O primeiro argumento fornecido identifica qual campo está sendo avaliado.

O segundo argumento é uma data codificada para 26 de julho de 2021, que é a `base_date` que identifica o limite final do segmento de ano. O `period_no` de 0 é o argumento final, o que significa que a função não está comparando anos anteriores ou posteriores ao ano segmentado.

Diagrama da função inyeartodate, sem argumentos adicionais



Qualquer transação que ocorra entre 1º e 26 de janeiro retorna um resultado booleano de TRUE. Datas de transações anteriores a 2021 e após 26 de julho de 2021 retornam FALSE.

Exemplo 2: period_no

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- O mesmo conjunto de dados e cenário do primeiro exemplo.
- A criação de um campo, `previous_year_to_date`, que determina quais transações ocorreram um ano inteiro antes do término do segmento de ano em 26 de julho de 2021.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

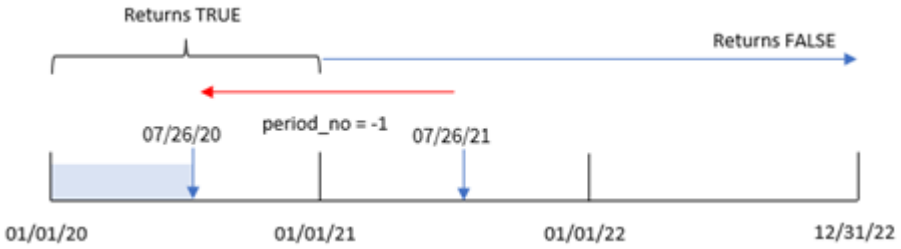
- `date`
- `previous_year_to_date`

Tabela de resultados

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

Um valor de `period_no` -1 indica que a função `inyeartodate ()` compara o segmento de trimestre de entrada com o ano anterior. Com uma data de entrada de 26 de julho de 2021, o segmento de 1º de janeiro de 2021 a 26 de julho de 2021 foi inicialmente identificado como o acumulado do ano. `period_no` então desloca esse segmento em um ano inteiro antes, fazendo com que os limites de data se tornem de 1º de janeiro a 26 de julho de 2020.

Diagrama da função inyeartodate, exemplo de period_no



Portanto, qualquer transação que ocorra entre 1º de janeiro e 26 de julho de 2020 retornará um resultado booleano de `TRUE`.

Exemplo 3: first_month_of_year

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- O mesmo conjunto de dados e cenário do primeiro exemplo.
- A criação de um campo, `in_year_to_date`, que determina quais transações ocorreram no mesmo ano até 26 de julho de 2021.

Neste exemplo, definimos março como o primeiro mês do ano fiscal.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `in_year_to_date`

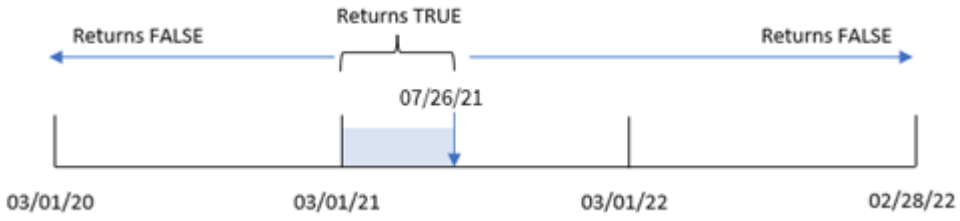
Tabela de resultados

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

date	in_year_to_date
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

Usando 3 como o argumento `first_month_of_year` na função `inyeartodate()`, a função começa o ano em 1º de março. A `base_date` de 26 de julho de 2021 define a data de término para esse segmento de ano.

Diagrama da função inyeartodate, exemplo de first_month_of_year



Portanto, qualquer transação que ocorra entre 1º de março e 26 de julho de 2021 retornará um resultado booleano de TRUE, enquanto as transações com datas fora desses limites retornarão um valor de FALSE.

Exemplo 4: Exemplo de objeto de gráfico

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém o mesmo conjunto de dados e cenário do primeiro exemplo. No entanto, neste exemplo, o conjunto de dados inalterado é carregado no aplicativo. O cálculo que determina quais transações ocorreram no mesmo ano até 26 de julho de 2021 é criado como uma medida em um objeto de gráfico no aplicativo.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão:`date`.

Crie a seguinte medida:

```
=inyeartodate(date,'07/26/2021', 0)
```

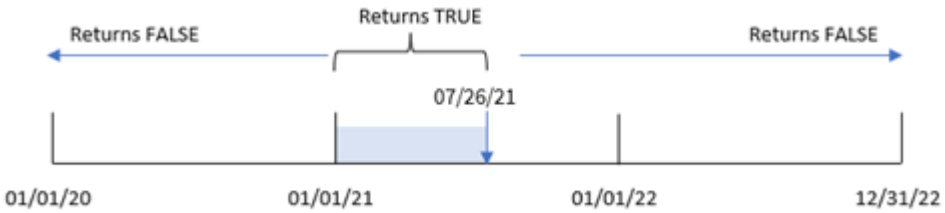
Tabela de resultados

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

date	=inyeartodate(date,'07/26/2021', 0)
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

A medida `in_year_to_date` é criada no objeto de gráfico usando a função `inyeartodate()`. O primeiro argumento fornecido identifica qual campo está sendo avaliado. O segundo argumento é uma data codificada para 26 de julho de 2021, que é a `base_date` que identifica o limite final do segmento de ano comparador. O `period_no` de 0 é o argumento final, o que significa que a função não está comparando anos anteriores ou posteriores ao ano segmentado.

Diagrama da função inyeartodate, exemplo de objeto de gráfico



Qualquer transação que ocorra entre 1º de janeiro e 26 de julho de 2021 retorna um resultado booleano de TRUE. Datas da transação antes de 2021 e depois de 26 de julho de 2021 retornam FALSE.

Exemplo 5: Cenário

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados que é carregado em uma tabela denominada `Products`.
- Informações sobre ID do produto, tipo de produto, data de fabricação e preço de custo.

O usuário final gostaria de um objeto de gráfico que exiba, por tipo de produto, o custo dos produtos fabricados em 2021 até 26 de julho.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão:`product_type`.

Crie uma medida que calcule a soma de cada produto que foi fabricado em 2021, antes de 27 de julho:

```
=sum(if(inyeartodate(manufacture_date,makedate(2021,07,26),0),cost_price,0))
```

Defina o **Formato numérico** da medida como **Dinheiro**.

Tabela de resultados

product_type	**=sum(if(inyeartodate(manufacture_date,makedate (2021,07,26),0),cost_price,0))**
produto A	$95.93
produto B	$128.66

product_type	=sum(if(inyeartodate(manufacture_date,makedate (2021,07,26),0),cost_price,0))
produto C	$61.89
produto D	$146.09

A função `inyeartodate()` retorna um valor booleano ao avaliar as datas de fabricação de cada um dos produtos. Para qualquer produto fabricado em 2021 antes de 27 de julho, a função `inyeartodate()` retorna um valor booleano de `TRUE` e soma o `cost_price`.

O produto D é o único que também foi fabricado após 26 de julho de 2021. A entrada com `product_ID` 8203 foi fabricada em 27 de dezembro e custou $ 25,12. Portanto, esse custo não foi incluído no total do Produto D no objeto de gráfico.

lastworkdate

A função **lastworkdate** retorna a primeira data de término para obter **no_of_workdays** (de segunda a sexta-feira), se começar em **start_date**, levando em consideração qualquer **holiday** opcionalmente listado. **start_date** e **holiday** devem ser datas ou carimbos de data/hora válidos.

Sintaxe:

```
lastworkdate(start_date, no_of_workdays {, holiday})
```

Tipo de dados de retorno: inteiro

Um calendário que mostra como a função `lastworkdate()` é usada

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			

Limitações

Não há nenhum método para modificar a função `lastworkdate()` para regiões ou cenários que envolvam algo diferente de uma semana de trabalho que começa na segunda-feira e termina na sexta-feira.

O parâmetro de feriado deve ser uma constante de cadeia de caracteres. Ele não aceita uma expressão.

Quando usar

A função `lastworkdate()` é comumente usada como parte de uma expressão quando o usuário deseja calcular a data de término proposta de um projeto ou tarefa, com base em quando o projeto começa e nos feriados que ocorrerão nesse período.

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Argumentos

Argumento	Descrição
start_date	A data inicial para avaliar.
no_of_ workdays	Número de dias úteis para alcançar.
holiday	Períodos de feriados a serem excluídos dos dias de trabalho. Um feriado é declarado como uma data constante de string. Você pode especificar várias datas de feriados, separadas por vírgulas. **Exemplo:** `'12/25/2013', '12/26/2013', '12/31/2013', '01/01/2014'`

Exemplo 1: Exemplo básico

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo IDs de projetos, datas de início do projeto e o esforço estimado, em dias, necessário para os projetos. O conjunto de dados é carregado em uma tabela denominada "`Projects`".
- Um carregamento anterior que contém a função `lastworkdate()` que é definida como o campo "`end_date`" e identifica quando cada projeto está programado para terminar.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `id`
- `start_date`
- `effort`
- `end_date`

Tabela de resultados

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

Como não há feriados programados, a função adiciona o número definido de dias úteis, de segunda a sexta-feira, à data de início para encontrar a data de término mais próxima possível.

O calendário a seguir mostra as datas de início e término do projeto 3, com os dias úteis destacados em verde.

Um calendário que mostra as datas de início e término do projeto 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				

Exemplo 2: Feriado único

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo IDs de projetos, datas de início do projeto e o esforço estimado, em dias, necessário para os projetos. O conjunto de dados é carregado em uma tabela denominada "Projects".
- Um carregamento anterior que contém a função `lastworkdate()` que é definida como o campo `end_date` e identifica quando cada projeto está programado para terminar.

No entanto, há um feriado agendado para 18 de maio de 2022. A função `lastworkdate()` na carga anterior inclui o feriado em seu terceiro argumento para identificar quando cada projeto está programado para terminar.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `id`
- `start_date`

- `effort`
- `end_date`

Tabela de resultados

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

O único feriado agendado é inserido como o terceiro argumento na função `lastworkdate()`. Como resultado, a data de término do projeto 3 é deslocada para um dia depois porque o feriado ocorre em um dos dias úteis antes da data de término.

O calendário a seguir mostra as datas de início e término do projeto 3 e mostra que o feriado altera a data de término do projeto em um dia.

Um calendário que mostra a data de início e término do projeto 3 com um feriado em 18 de maio

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				

Exemplo 3: Vários feriados

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo IDs de projetos, datas de início do projeto e o esforço estimado, em dias, necessário para os projetos. O conjunto de dados é carregado em uma tabela denominada "`Projects`".
- Um carregamento anterior que contém a função `lastworkdate()` que é definida como o campo `end_date` e identifica quando cada projeto está programado para terminar.

No entanto, há três feriados programados para 19, 20, 21 e 22 de maio. A função `lastworkdate()` no carregamento anterior inclui cada um dos feriados em seu terceiro argumento para identificar quando cada projeto está programado para terminar.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `id`
- `start_date`

- effort
- end_date

Tabela de resultados

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

Os quatro feriados são inseridos como uma lista de argumentos na função `lastworkdate()` após a data de início e o número de dias úteis.

O calendário a seguir mostra as datas de início e término do projeto 3 e mostra que os feriados alteram a data de término do projeto em três dias.

Um calendário que mostra a data de início e término do projeto 3 com feriados de 19 a 22 de maio

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				

Exemplo 4: Feriado único (gráfico)

Script de carregamento e expressão de gráfico

Visão geral

São usados o mesmo conjunto de dados e cenário do primeiro exemplo.

No entanto, neste exemplo, o conjunto de dados permanece inalterado e carregado no aplicativo. O campo `end_date` é calculado como uma medida em um gráfico.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `id`
- `start_date`
- `effort`

Para calcular end_date, crie a seguinte medida:

- `=LastWorkDate(start_date,effort,'05/18/2022')`

Tabela de resultados

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

id	start_date	effort	=LastWorkDate(start_date,effort,'05/18/2022')
5	08/10/2022	26	09/14/2022

O único feriado programado é inserido como uma medida no gráfico. Como resultado, a data de término do projeto 3 é deslocada para um dia depois porque o feriado ocorre em um dos dias úteis antes da data de término.

O calendário a seguir mostra as datas de início e término do projeto 3 e mostra que o feriado altera a data de término do projeto em um dia.

Um calendário que mostra a data de início e término do projeto 3 com um feriado em 18 de maio

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

Essa função retorna um carimbo de data/hora da hora atual para um fuso horário especificado.

Sintaxe:

```
LocalTime([timezone [, ignoreDST ]])
```

Tipo de dados de retorno: dual

Argumentos

Argumento	Descrição
timezone	A **timezone** é especificada como uma string que contém qualquer um dos locais geográficos listados em **Fuso Horário**, no **Painel de Controle do Windows**, para **Data e Hora** ou como uma string no formato "GMT+hh:mm". Uma lista de locais e fusos horários aceitos também é apresentada na tabela abaixo. Se nenhum fuso horário for especificado, a hora local será retornada. *Se você usar um deslocamento de horário de verão (ou seja, especificar um valor de argumento **ignoreDST** avaliando como False), deverá especificar um local, em vez de um deslocamento GMT, no argumento **place**. Isso ocorre porque o ajuste para o horário de verão requer informações latitudinais além das informações longitudinais fornecidas por um deslocamento GMT. Para obter mais informações, consulte Usando deslocamentos GMT em combinação com DST (page 854).*
ignoreDST	Se este argumento for avaliado como True, DST (horário de verão) será ignorado. Valores de argumentos válidos avaliados para True incluem `-1` e `True()`. Se esse argumento for avaliado como False, o carimbo de data/hora será ajustado para o horário de verão. Valores de argumentos válidos avaliados para False incluem `0` e `False()`. Se o valor do argumento **ignoreDST** for inválido, a função avaliará a expressão como se o valor **ignore_dst** fosse avaliado como True. Se o valor do argumento **ignoreDST** não for especificado, a função avaliará a expressão como se o valor **ignore_dst** fosse avaliado como False.

Locais e fusos horários válidos

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	-	-	-

Exemplos e resultados:

Os exemplos abaixo são baseados na função sendo chamada em 2023-08-14 08:39:47 hora local, com o fuso horário local do servidor ou ambiente de desktop sendo GMT-05:00 e em uma região que implementou a luz do dia economizando tempo a partir desta data listada.

Exemplos de scripts

Exemplo	Resultado
`localtime ()`	Retorna a hora local 2023-08-14 08:39:47.
`localtime ('London')`	Retorna a hora local em Londres, 2023-08-14 13:39:47.
`localtime ('GMT+02:00')`	Retorna a hora local no fuso horário GMT+02:00, 2023-08-14 14:39:47. Nenhum ajuste é feito para o horário de verão porque um deslocamento GMT, em vez de um local, é especificado.
`localtime ('Paris',-1)`	Retorna a hora local em Paris, sem considerar o horário de verão, 2023-08-14 13:39:47.
`localtime ('Paris',True())`	Retorna a hora local em Paris, sem considerar o horário de verão, 2023-08-14 13:39:47.
`localtime ('Paris',0)`	Retorna a hora local de Paris, considerando o horário de verão, 2023-08-14 14:39:47.
`localtime ('Paris',False ())`	Retorna a hora local de Paris, considerando o horário de verão, 2023-08-14 14:39:47.

Usando deslocamentos GMT em combinação com DST

Após a implementação das bibliotecas de Componentes Internacionais para Unicode (ICU) no Qlik Sense, o uso de deslocamentos GMT (Greenwich Mean Time) em combinação com DST (Daylight Saving Time) requer informações latitudinais adicionais.

GMT é um deslocamento longitudinal (leste-oeste), enquanto DST é um deslocamento latitudinal (norte-sul). Por exemplo, Helsinque (Finlândia) e Joanesburgo (África do Sul) compartilham o mesmo deslocamento GMT+02:00, mas não compartilham o mesmo deslocamento DST. Isso significa que, além do deslocamento GMT, qualquer deslocamento DST requer informações sobre a posição latitudinal do fuso horário local (entrada de fuso horário geográfico) para obter informações completas sobre as condições locais do DST.

lunarweekend

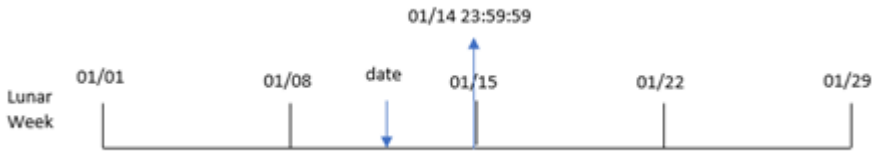
Essa função retorna um valor correspondente a um carimbo de data/hora do último milissegundo do último dia da semana lunar que contém **date**. As semanas lunares no Qlik Sense são definidas contando 1º de janeiro como o primeiro dia da semana e, além da última semana do ano, conterão exatamente sete dias.

Sintaxe:

```
LunarweekEnd(date[, period_no[, first_week_day]])
```

Tipo de dados de retorno: dual

Exemplo de diagrama da função `lunarweekend()`



A função `lunarweekend()` determina em qual semana lunar a `date` cai. Em seguida, ela retorna um carimbo de data/hora, em formato de data, para o último milissegundo daquela semana.

Argumentos

Argumento	Descrição
date	A data ou o carimbo de data/hora a ser avaliado.
period_no	**period_no** é um inteiro ou uma expressão que resolve um inteiro em que o valor 0 indica a semana lunar que contém **date**. Valores negativos em **period_no** indicam semanas lunares precedentes e valores positivos indicam semanas lunares sucessivas.
first_week_day	Um deslocamento que pode ser maior ou menor que zero. Isso muda o início do ano por um número determinado de dias e/ou frações de um dia.

Quando usar

A função `lunarweekend()` é comumente usada como parte de uma expressão quando o usuário deseja que o cálculo use a fração da semana que ainda não ocorreu. Diferente da função `weekend()`, a última semana lunar de cada ano civil terminará em 31 de dezembro. Por exemplo, a função `lunarweekend()` pode ser usada para calcular juros ainda não acumulados durante a semana.

Exemplos de funções

Exemplo	Resultado
`lunarweekend('01/12/2013')`	Retorna 01/14/2013 23:59:59.
`lunarweekend('01/12/2013', -1)`	Retorna 01/07/2013 23:59:59.
`lunarweekend('01/12/2013', 0, 1)`	Retorna 01/15/2013 23:59:59.

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplo 1: Sem argumentos adicionais

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações para 2022, que é carregado em uma tabela denominada `Transactions`.
- O campo de data fornecido no formato da `variável` de sistema DateFormat (MM/DD/AAAA).
- A criação de um campo, `end_of_week`, que retorna um carimbo de data/hora para o final da semana lunar em que as transações ocorreram.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `end_of_week`
- `end_of_week_timestamp`

Tabela de resultados

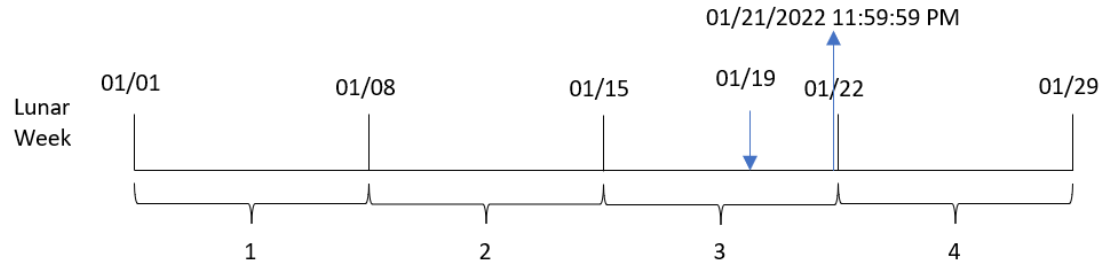
date	end_of_week	end_of_week_timestamp
1/7/2022	01/07/2022	1/7/2022 11:59:59 PM
1/19/2022	01/21/2022	1/21/2022 11:59:59 PM
2/5/2022	02/11/2022	2/11/2022 11:59:59 PM
2/28/2022	03/04/2022	3/4/2022 11:59:59 PM
3/16/2022	03/18/2022	3/18/2022 11:59:59 PM
4/1/2022	04/01/2022	4/1/2022 11:59:59 PM
5/7/2022	05/13/2022	5/13/2022 11:59:59 PM
5/16/2022	05/20/2022	5/20/2022 11:59:59 PM
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

date	end_of_week	end_of_week_timestamp
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

O campo `end_of_week` é criado na instrução de carregamento anterior usando a função `lunarweekend()` e transmitindo o campo `date` como o argumento da função.

A função `lunarweekend()` identifica em qual semana lunar o valor da data cai, retornando um carimbo de data/hora para o último milissegundo dessa semana.

Diagrama da função `lunarweekend()`, exemplo sem argumentos adicionais



A transação 8189 ocorreu em 19 de janeiro. A função `lunarweekend()` identifica que a semana lunar começa em 15 de janeiro. Portanto, o valor `end_of_week` dessa transação retorna o último milissegundo da semana lunar, que é 21 de janeiro às 11:59:59 PM.

Exemplo 2: period_no

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- O mesmo conjunto de dados e cenário do primeiro exemplo.
- A criação de um campo, `previous_lunar_week_end`, que retorna o carimbo de data/hora do final da semana lunar antes da transação.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- date
- previous_lunar_week_end
- previous_lunar_week_end_timestamp

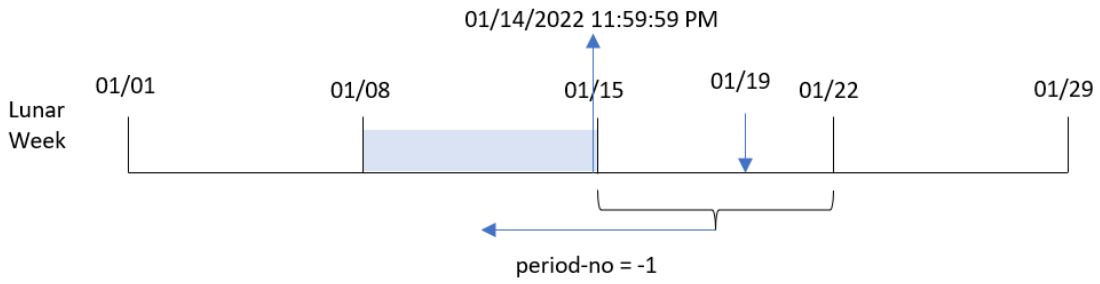
Tabela de resultados

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

date	previous_lunar_week_end	previous_lunar_week_end_timestamp
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

Nesse caso, como um `period_no` de -1 foi usado como argumento de compensação na função `lunarweekend()`, a função primeiro identifica a semana lunar em que as transações ocorreram. Em seguida, ela muda para uma semana antes e identifica o último milissegundo dessa semana lunar.

Diagrama da função `lunarweekend()`, exemplo de period_no



A transação 8189 ocorreu em 19 de janeiro. A função `lunarweekend()` identifica que a semana lunar começa em 15 de janeiro. Portanto, a semana lunar anterior começou em 8 de janeiro e terminou em 14 de janeiro às 11:59:59 PM. Esse é o valor retornado para o campo `previous_lunar_week_end`.

Exemplo 3: first_week_day

Script de carregamento e resultados

Visão geral

Abra o editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém o mesmo conjunto de dados e cenário do primeiro exemplo. Neste exemplo, definimos que as semanas lunares devem começar em 5 de janeiro.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

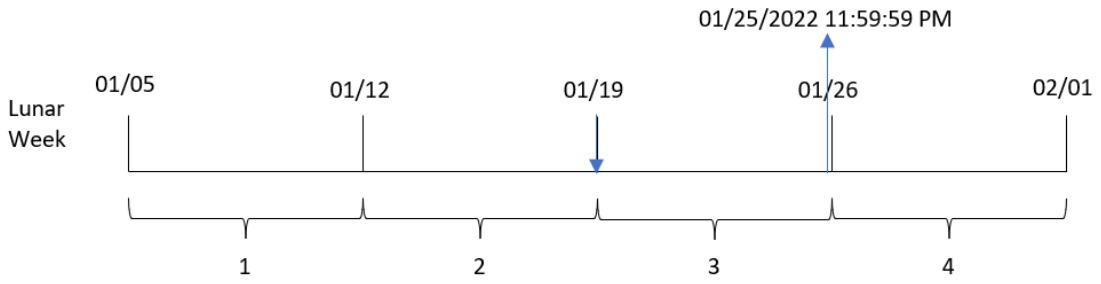
- `date`
- `end_of_week`
- `end_of_week_timestamp`

Tabela de resultados

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
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

Nesse caso, como o argumento `first_week_date` de 4 é usado na função `lunarweekend()`, ele desloca o início do ano de 1º de janeiro a 5 de janeiro.

Diagrama da função `lunarweekend()`, exemplo de first_week_day



A transação 8189 ocorreu em 19 de janeiro. Como as semanas lunares começam em 5 de janeiro, a função `lunarweekend()` identifica que a semana lunar contendo 19 de janeiro também começa em 19 de janeiro. Portanto, o final dessa semana lunar ocorre em 25 de janeiro às 11:59:59 PM; esse é o valor retornado para o campo `end_of_week`.

Exemplo 4: Exemplo de objeto de gráfico

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém o mesmo conjunto de dados e cenário do primeiro exemplo.

No entanto, neste exemplo, o conjunto de dados inalterado é carregado no aplicativo. O cálculo que retorna um carimbo de data/hora para o final da semana lunar em que as transações ocorreram é criado como uma medida em um objeto de gráfico do aplicativo.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão: `date`.

Adicione as seguintes medidas:

```
=lunarweekend(date)
```

```
=timestamp(lunarweekend(date))
```

Tabela de resultados

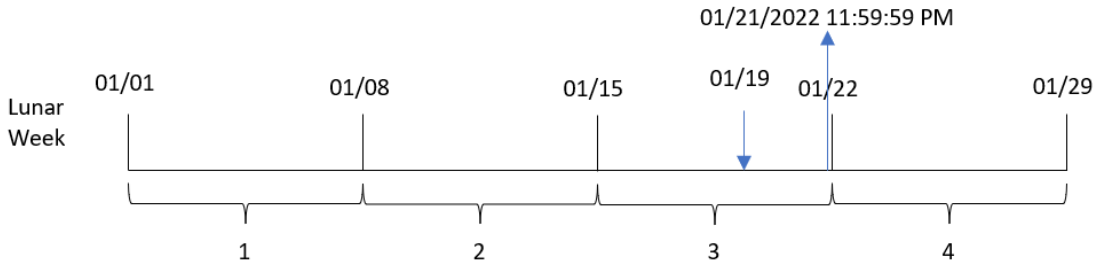
date	=lunarweekend(date)	=timestamp(lunarweekend(date))
1/7/2022	01/07/2022	1/7/2022 11:59:59 PM
1/19/2022	01/21/2022	1/21/2022 11:59:59 PM
2/5/2022	02/11/2022	2/11/2022 11:59:59 PM
2/28/2022	03/04/2022	3/4/2022 11:59:59 PM
3/16/2022	03/18/2022	3/18/2022 11:59:59 PM
4/1/2022	04/01/2022	4/1/2022 11:59:59 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
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

date	=lunarweekend(date)	=timestamp(lunarweekend(date))
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

A medida `end_of_week` é criada no objeto de gráfico usando a função `lunarweekend()` e transmitindo o campo `date` como o argumento da função.

A função `lunarweekend()` identifica em qual semana lunar o valor da data cai, retornando um carimbo de data/hora para o último milissegundo dessa semana.

Diagrama da função `lunarweekend()`, exemplo de objeto de gráfico



A transação 8189 ocorreu em 19 de janeiro. A função `lunarweekend()` identifica que a semana lunar começa em 15 de janeiro. Portanto, o valor `end_of_week` dessa transação retorna o último milissegundo da semana lunar, que é 21 de janeiro às 11:59:59 PM.

Exemplo 5: Cenário

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados que é carregado em uma tabela denominada `Employee_Expenses`.
- Os IDs dos funcionários, o nome do funcionário e a média diária de reivindicações de despesas de cada funcionário.

O usuário final deseja um objeto de gráfico que mostre, por ID de funcionário e nome de funcionário, as reivindicações de despesas estimadas ainda a serem acumuladas para o restante da semana lunar.

Script de carregamento

```
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
];
```

Resultados

Faça o seguinte:

1. Carregue os dados e abra uma pasta. Crie uma nova tabela.
2. Adicione os seguintes campos como dimensões:
   - `employee_id`
   - `employee_name`
3. Em seguida, crie a seguinte medida para calcular os juros acumulados:
   `=(lunarweekend(today(1))-today(1))*avg_daily_claim`
4. Defina o **Formato numérico** da medida como **Dinheiro**.

Tabela de resultados

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

A função `lunarkweekend()`, usando a data de hoje como seu único argumento, retorna a data de término da semana lunar atual. Em seguida, subtraindo a data de hoje da data de término da semana lunar, a expressão retorna o número de dias que restam nessa semana.

Esse valor é então multiplicado pela média de solicitações de despesas diárias por cada funcionário para calcular o valor estimado das solicitações que cada funcionário deve fazer na semana lunar restante.

lunarweekname

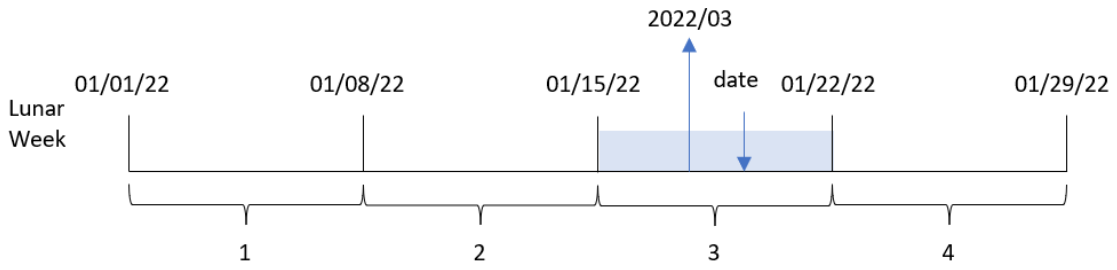
Esta função retorna um valor de exibição que mostra o número do ano e da semana lunar que corresponde a data/hora do primeiro milissegundo do primeiro dia da semana lunar que contém a **date**. As semanas lunares no Qlik Sense são definidas contando 1º de janeiro como o primeiro dia da semana e, além da última semana do ano, conterão exatamente sete dias.

Sintaxe:

```
LunarWeekName(date [, period_no[, first_week_day]])
```

Tipo de dados de retorno: dual

Exemplo de diagrama da função `lunarweekname()`



A função `lunarweekname()` determina em qual semana lunar a data cai, começando uma contagem de semanas a partir de 1º de janeiro. Em seguida, ela retorna um valor composto por `year/weekcount`.

Argumentos

Argumento	Descrição
date	A data ou o carimbo de data/hora a ser avaliado.
period_no	**period_no** é um inteiro ou uma expressão que resolve um inteiro em que o valor 0 indica a semana lunar que contém **date**. Valores negativos em **period_no** indicam semanas lunares precedentes e valores positivos indicam semanas lunares sucessivas.
first_week_day	Um deslocamento que pode ser maior ou menor que zero. Isso muda o início do ano por um número determinado de dias e/ou frações de um dia.

Quando usar

A função `lunarweekname()` é útil quando você deseja comparar agregações por semanas lunares. Por exemplo, a função pode ser usada para determinar o total de vendas de produtos por semana lunar. Semanas lunares são úteis quando você deseja garantir que todos os valores contidos na primeira semana do ano contenham apenas valores a partir de 1º de janeiro, no mínimo.

Essas dimensões podem ser criadas no script de carregamento usando a função para criar um campo em uma tabela de Calendário mestre. A função também pode ser usada diretamente em um gráfico como uma dimensão calculada.

Exemplos de funções

Exemplo	Resultado
`lunarweekname('01/12/2013')`	Retorna 2006/02.
`lunarweekname('01/12/2013', -1)`	Retorna 2006/01.
`lunarweekname('01/12/2013', 0, 1)`	Retorna 2006/02.

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplo 1: data sem argumentos adicionais

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações para 2022, que é carregado em uma tabela denominada `Transactions`.
- O campo de data fornecido no formato da `variável` de sistema DateFormat (MM/DD/AAAA).
- A criação de um campo, `lunar_week_name`, que retorna o número do ano e da semana lunar em que as transações ocorreram.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `lunar_week_name`

Tabela de resultados

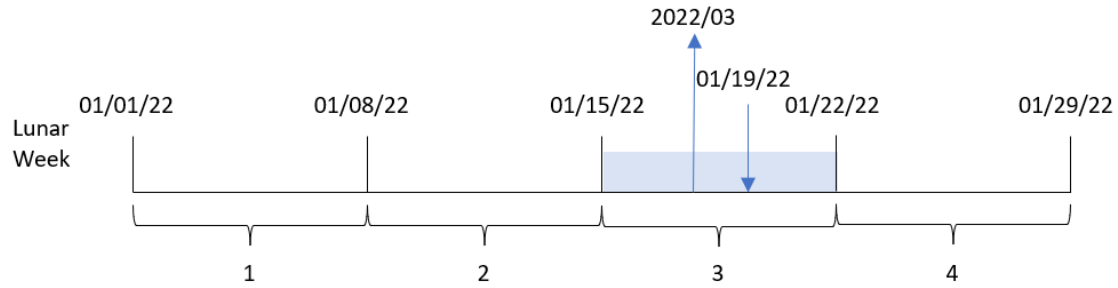
date	lunar_week_name
1/7/2022	2022/01
1/19/2022	2022/03
2/5/2022	2022/06
2/28/2022	2022/09
3/16/2022	2022/11
4/1/2022	2022/13
5/7/2022	2022/19
5/16/2022	2022/20
6/15/2022	2022/24
6/26/2022	2022/26
7/9/2022	2022/28
7/22/2022	2022/29
7/23/2022	2022/30
7/27/2022	2022/30

date	lunar_week_name
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

O campo `lunar_week_name` é criado na instrução de carregamento anterior usando a função `lunarweekname()` e transmitindo o campo `date` como o argumento da função.

A função `lunarweekname()` identifica em qual semana lunar o valor da data cai, retornando o número do ano e da semana dessa data.

Diagrama da função `lunarweekname()`, exemplo sem argumentos adicionais



A transação 8189 ocorreu em 19 de janeiro. A função `lunarweekname()` identifica que essa data cai na semana lunar que começa em 15 de janeiro; essa é a terceira semana lunar do ano. Portanto, o valor `lunar_week_name` retornado para essa transação é 2022/03.

Exemplo 2: data com o argumento period_no

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- O mesmo conjunto de dados e cenário do primeiro exemplo.
- A criação de um campo, `previous_lunar_week_name`, que retorna o número do ano e da semana da semana lunar anterior à data em que as transações ocorreram.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `previous_lunar_week_name`

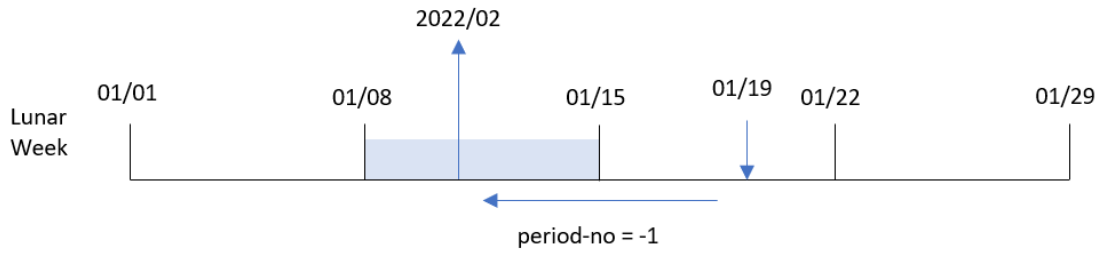
Tabela de resultados

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

date	previous_lunar_week_name
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

Nesse caso, como um `period_no` de -1 foi usado como argumento de compensação na função `lunarweekname()`, a função primeiro identifica a semana lunar em que as transações ocorreram. Em seguida, ele retorna o ano e o número de uma semana antes.

Diagrama da função `lunarweekname()`, exemplo de period_no



A transação 8189 ocorreu em 19 de janeiro. A função `lunarweekname()` identifica que essa transação ocorreu na terceira semana lunar do ano. Portanto, ela retorna o ano e o valor de uma semana antes, 2022/02, para o campo `previous_lunar_week_name`.

Exemplo 3: data com o argumento first_week_day

Script de carregamento e resultados

Visão geral

Abra o editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém o mesmo conjunto de dados e cenário do primeiro exemplo. Neste exemplo, definimos que as semanas lunares devem começar em 5 de janeiro.

Script de carregamento

```
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
];
```

Resultados

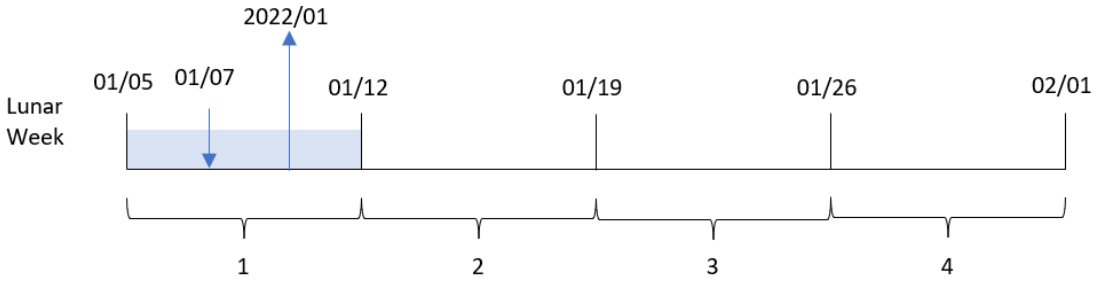
Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `lunar_week_name`

Tabela de resultados

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
10/14/2022	2022/41
10/29/2022	2022/43

Diagrama da função `lunarweekname()`, exemplo de first_week_day



Nesse caso, como o argumento `first_week_date` de 4 é usado na função `lunarweekname()`, ele desloca o início das semanas lunares de 1º de janeiro a 5 de janeiro.

A transação 8188 ocorreu em 7 de janeiro. Como as semanas lunares começam em 5 de janeiro, a função `lunarweekname()` identifica que a semana lunar contendo 7 de janeiro é a primeira semana lunar do ano. Portanto, o valor de `lunar_week_name` retornado para essa transação é 2022/01.

Exemplo 4: Exemplo de objeto de gráfico

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém o mesmo conjunto de dados e cenário do primeiro exemplo.

No entanto, neste exemplo, o conjunto de dados inalterado é carregado no aplicativo. O cálculo que retorna o número da semana lunar e o ano em que as transações ocorreram é criado como uma medida em um objeto de gráfico do aplicativo.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão: `date`.

Para calcular a data de início da semana lunar em que uma transação ocorre, crie a seguinte medida:

```
=lunarweekname(date)
```

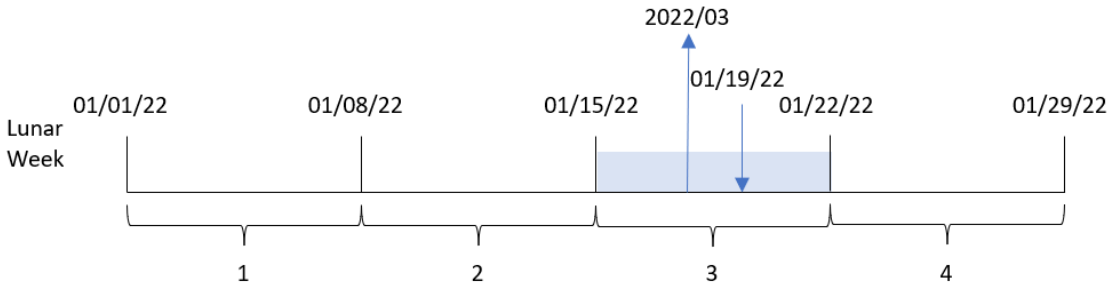
Tabela de resultados

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
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

A medida `lunar_week_name` é criada no objeto de gráfico usando a função `lunarweekname()` e transmitindo o campo `date` como o argumento da função.

A função `lunarweekname()` identifica em qual semana lunar o valor da data cai, retornando o número do ano e da semana dessa data.

Diagrama da função `lunarweekname()`, exemplo de objeto de gráfico



A transação 8189 ocorreu em 19 de janeiro. A função `lunarweekname()` identifica que essa data cai na semana lunar que começa em 15 de janeiro; essa é a terceira semana lunar do ano. Portanto, o valor de `lunar_week_name` dessa transação é 2022/03.

Exemplo 5: Cenário

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações para 2022, que é carregado em uma tabela chamada `Transactions`.
- O campo de data fornecido no formato da variável de sistema `DateFormat` (MM/DD/AAAA).

O usuário final gostaria de um objeto de gráfico que apresentasse o total de vendas por semana no ano atual. A semana 1, com duração de sete dias, deve começar em 1º de janeiro. Isso pode ser alcançado mesmo quando essa dimensão não está disponível no modelo de dados. Para isso, use a função `lunarweekname()` como uma dimensão calculada no gráfico.

Script de carregamento

```
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
];
```

Resultados

Faça o seguinte:

1. Carregue os dados e abra uma pasta. Crie uma nova tabela.
2. Crie uma dimensão calculada usando a seguinte expressão:
   `=lunarweekname(date)`
3. Calcule o total de vendas usando a seguinte medida de agregação:
   `=sum(amount)`
4. Defina o **Formato numérico** da medida como **Dinheiro**.

Tabela de resultados

=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

=lunarweekname(date)	=sum(amount)
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

Essa função retorna um valor correspondente a um carimbo de data/hora do primeiro milissegundo do primeiro dia da semana lunar que contém **date**. As semanas lunares no Qlik Sense são definidas contando 1º de janeiro como o primeiro dia da semana e, além da última semana do ano, conterão exatamente sete dias.

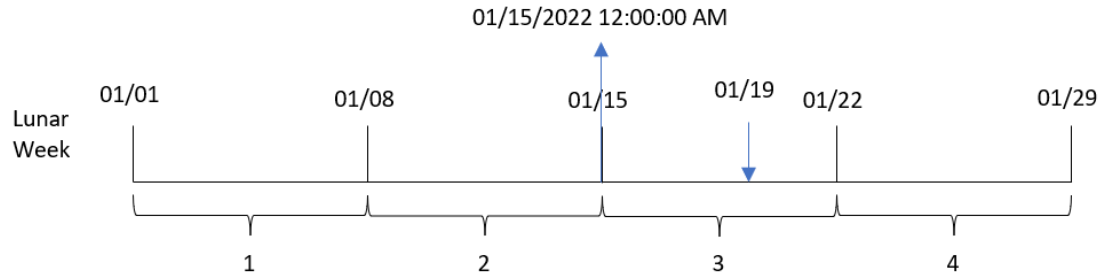
Sintaxe:

```
LunarweekStart(date[, period_no[, first_week_day]])
```

Tipo de dados de retorno: dual

A função `lunarweekstart()` determina em qual semana lunar a `date` cai. Em seguida, ela retorna um carimbo de data/hora, em formato de data, para o primeiro milissegundo daquela semana.

Exemplo de diagrama da função `lunarweekstart()`



Argumentos

Argumento	Descrição
date	A data ou o carimbo de data/hora a ser avaliado.
period_no	**period_no** é um inteiro ou uma expressão que resolve um inteiro em que o valor 0 indica a semana lunar que contém **date**. Valores negativos em **period_no** indicam semanas lunares precedentes e valores positivos indicam semanas lunares sucessivas.
first_week_day	Um deslocamento que pode ser maior ou menor que zero. Isso muda o início do ano por um número determinado de dias e/ou frações de um dia.

Quando usar

A função `lunarweekstart()` é comumente usada como parte de uma expressão quando o usuário deseja que o cálculo use a fração da semana decorrida até o momento. Diferentemente da função `weekstart()`, no início de cada novo ano civil, a semana começa em 1º de janeiro e cada semana subsequente começa sete dias depois. A função `lunarweekstart()` não é afetada pela `variável` de sistema FirstWeekDay.

Por exemplo, `lunarweekstart()` pode ser usado para calcular os juros acumulados em uma semana até o momento.

Exemplos de funções

Exemplo	Resultado
`lunarweekstart ('01/12/2013')`	Retorna 01/08/2013.
`lunarweekstart ('01/12/2013', -1)`	Retorna 01/01/2013.
`lunarweekstart ('01/12/2013', 0, 1 )`	Retorna 01/09/2013, porque definir first_week_day como 1 significa que o início do ano foi alterado para 01/02/2013.

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplo 1: sem argumentos adicionais

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações para 2022, que é carregado em uma tabela denominada `Transactions`.
- O campo de data fornecido no formato da variável de sistema `DateFormat` (MM/DD/AAAA).
- A criação de um campo, `start_of_week`, que retorna um carimbo de data/hora para o início da semana lunar em que as transações ocorreram.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `start_of_week`
- `start_of_week_timestamp`

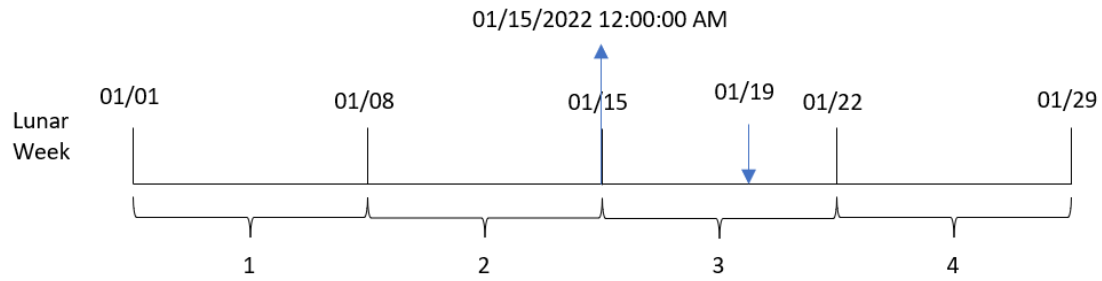
Tabela de resultados

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
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

O campo `start_of_week` é criado na instrução de carregamento anterior usando a função `lunarweekstart()` e transmitindo o campo `date` como o argumento da função.

A função `lunarweekstart()` identifica a semana lunar na qual a data cai, retornando um carimbo de data/hora para o primeiro milissegundo dessa semana.

Diagrama da função `lunarweekstart()`, exemplo sem argumentos adicionais



A transação 8189 ocorreu em 19 de janeiro. A função `lunarweekstart()` identifica que a semana lunar começa em 15 de janeiro. Portanto, o valor `start_of_week` dessa transação retorna o primeiro milissegundo desse dia, que é 15 de janeiro às 12:00:00 AM.

Exemplo 2: period_no

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- O mesmo conjunto de dados e cenário do primeiro exemplo.
- A criação de um campo `previous_lunar_week_start`, que retorna o carimbo de data/hora do início da semana antes da transação.

Script de carregamento

```
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
];
```

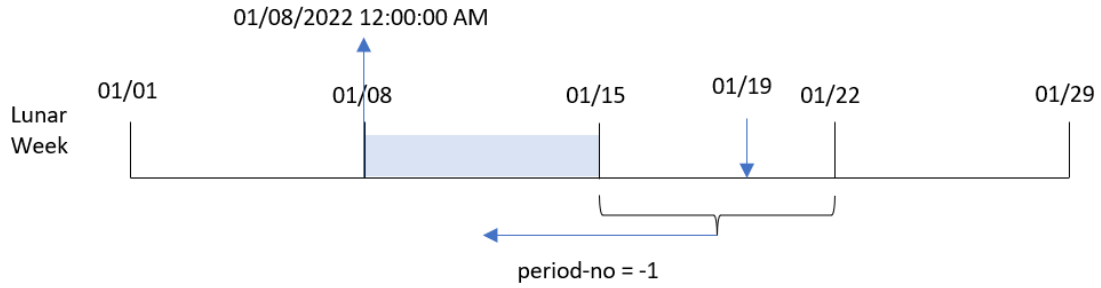
Resultados

Tabela de resultados

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
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

Nesse caso, como um `period_no` de -1 foi usado como argumento de deslocamento na função `lunarweekstart()`, a função primeiro identifica a semana lunar em que as transações ocorrem. Em seguida, ela muda uma semana antes e identifica o primeiro milissegundo dessa semana lunar.

Diagrama da função `lunarweekstart()`, exemplo de period_no



A transação 8189 ocorreu em 19 de janeiro. A função `lunarweekstart()` identifica que a semana lunar começa em 15 de janeiro. Portanto, a semana lunar anterior começou em 8 de janeiro às 12h00; esse é o valor retornado para o campo `previous_lunar_week_start`.

Exemplo 3: first_week_day

Script de carregamento e resultados

Visão geral

Abra o editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém o mesmo conjunto de dados e cenário do primeiro exemplo. Neste exemplo, definimos que as semanas lunares devem começar em 5 de janeiro.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `start_of_week`
- `start_of_week_timestamp`

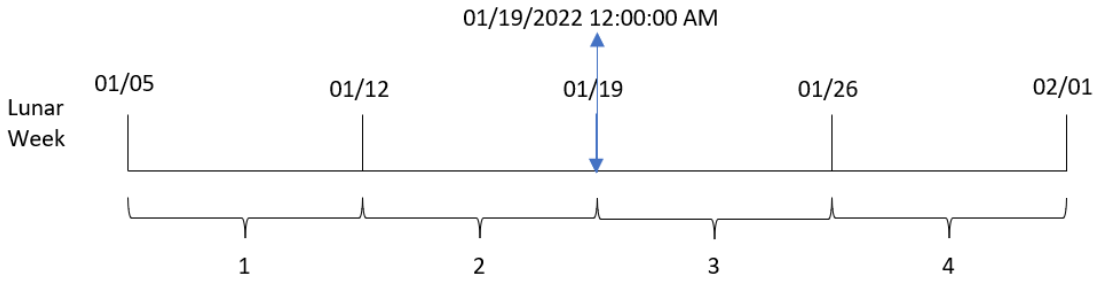
Tabela de resultados

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

date	start_of_week	start_of_week_timestamp
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

Nesse caso, como o argumento `first_week_date` de 4 é usado na função `lunarweekstart()`, ele desloca o início do ano de 1º de janeiro a 5 de janeiro.

Diagrama da função `lunarweekstart()`, exemplo de first_week_day



A transação 8189 ocorreu em 19 de janeiro. Como as semanas lunares começam em 5 de janeiro, a função `lunarweekstart()` identifica que a semana lunar contendo 19 de janeiro também começa em 19 de janeiro às 12:00:00 AM. Portanto, esse é o valor retornado para o campo `start_of_week`.

Exemplo 4: Exemplo de objeto de gráfico

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém o mesmo conjunto de dados e cenário do primeiro exemplo.

No entanto, neste exemplo, o conjunto de dados inalterado é carregado no aplicativo. O cálculo que retorna um carimbo de data/hora para o início da semana lunar em que as transações ocorreram é criado como uma medida em um objeto de gráfico do aplicativo.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão: `date`.

Adicione as seguintes medidas:

```
=lunarweekstart(date)
```

```
=timestamp(lunarweekstart(date))
```

Tabela de resultados

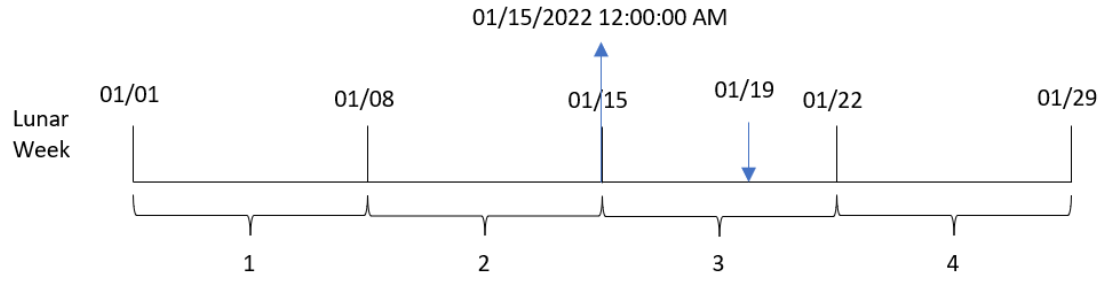
date	=lunarweekstart(date)	=timestamp(lunarweekstart(date))
1/7/2022	01/01/2022	1/1/2022 12:00:00 AM
1/19/2022	01/15/2022	1/15/2022 12:00:00 AM
2/5/2022	02/05/2022	2/5/2022 12:00:00 AM
2/28/2022	02/26/2022	2/26/2022 12:00:00 AM
3/16/2022	03/12/2022	3/12/2022 12:00:00 AM
4/1/2022	03/26/2022	3/26/2022 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

date	=lunarweekstart(date)	=timestamp(lunarweekstart(date))
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
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

A medida `start_of_week` é criada no objeto de gráfico usando a função `lunarweekstart()` e transmitindo o campo de data como o argumento da função.

A função `lunarweekstart()` identifica em qual semana lunar o valor da data cai, retornando um carimbo de data/hora para o último milissegundo dessa semana.

Diagrama da função `lunarweekstart()`, exemplo de objeto de gráfico



A transação 8189 ocorreu em 19 de janeiro. A função `lunarweekstart()` identifica que a semana lunar começa em 15 de janeiro. Portanto, o valor de `start_of_week` dessa transação é o primeiro milissegundo desse dia, que é 15 de janeiro às 12h00.

Exemplo 5: Cenário

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de saldos de empréstimos, que é carregado em uma tabela denominada `Loans`.
- Dados que consistem em IDs de empréstimos, saldo no início da semana e taxa de juros simples cobrada em cada empréstimo por ano.

O usuário final deseja um objeto de gráfico que mostre, por ID de empréstimo, os juros atuais que foram acumulados em cada empréstimo na semana até o momento.

Script de carregamento

```
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
];
```

Resultados

Faça o seguinte:

1. Carregue os dados e abra uma pasta. Crie uma nova tabela.
2. Adicione os seguintes campos como dimensões.
    - `loan_id`
    - `start_balance`
3. Em seguida, crie a seguinte medida para calcular os juros acumulados:
   `=start_balance*(rate*(today(1)-lunarweekstart(today(1)))/365)`
4. Defina o **Formato numérico** da medida como **Dinheiro**.

Tabela de resultados

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

A função `lunarweekstart()`, usando a data de hoje como seu único argumento, retorna a data de início do ano atual. Ao subtrair esse resultado da data atual, a expressão retorna o número de dias decorridos até o momento nesta semana.

Esse valor é então multiplicado pela taxa de juros e dividido por 365 para retornar a taxa de juros efetiva acumulada do período. O resultado é então multiplicado pelo saldo inicial do empréstimo para retornar os juros acumulados até o momento nesta semana.

makedate

Esta função retorna uma data calculada a partir do ano **YYYY**, do mês **MM** e do dia **DD**.

Sintaxe:

```
MakeDate(YYYY [ , MM [ , DD ] ])
```

Tipo de dados de retorno: dual

Argumentos

Argumento	Descrição
YYYY	O ano como um inteiro.
MM	O mês como um inteiro. Se nenhum mês for indicado, 1 (janeiro) será assumido.
DD	O dia como um inteiro. Se nenhum dia for indicado, 1 (o primeiro) será assumido.

Quando usar

A função `makedate()` normalmente seria usada no script para geração de dados para gerar um calendário. Isso também pode ser usado quando o campo de data não está diretamente disponível como data, mas precisa de algumas transformações para extrair componentes de ano, mês e dia.

Esses exemplos usam o formato de data MM/DD/AAAA. O formato de data é especificado no comando `SET DateFormat` na parte superior do seu script de carregamento de dados. Altere o formato nos exemplos para atender às suas necessidades.

Exemplos de funções

Exemplo	Resultado
`makedate(2012)`	Retorna 01/01/2012.
`makedate(12)`	Retorna 01/01/2012.
`makedate(2012,12)`	Retorna 12/01/2012.
`makedate(2012,2,14)`	Retorna 02/14/2012.

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações

regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplo 1: Exemplo básico

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações para 2018, que é carregado em uma tabela denominada `Transactions`.
- O campo de data fornecido no formato da `variável` de sistema DateFormat (MM/DD/AAAA).
- A criação de um campo, `transaction_date`, que retorna uma data no formato MM/DD/AAAA.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- transaction_year
- transaction_month
- transaction_day
- transaction_date

Tabela de resultados

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

O campo transaction_date é criado na instrução de carregamento anterior usando a função makedate() e informando os campos ano, mês e dia como argumentos da função.

A função então combina e converte esses valores em um campo de data, retornando os resultados no formato da variável de sistema DateFormat.

Exemplo 2: Formato de data modificado

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- O mesmo conjunto de dados e cenário do primeiro exemplo.
- A criação de um campo, transaction_date, no formato DD/MM/AAAA sem modificar a variável de sistema DateFormat.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `transaction_year`
- `transaction_month`
- `transaction_day`
- `transaction_date`

Tabela de resultados

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

Nesse caso, a função `makedate()` está aninhada dentro da função `date()`. O segundo argumento da função `date()` define o formato dos resultados da função `makedate()` como o DD/MM/AAAA necessário.

Exemplo 3: Exemplo de objeto de gráfico

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações para 2018, que é carregado em uma tabela chamada `Transactions`.
- As datas da transação são fornecidas em dois campos: `year` e `month`.

Crie uma medida de objeto de gráfico, `transaction_date`, que retorna uma data no formato MM/DD/AAAA.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `year`
- `month`

Para determinar a `transaction_date`, crie essa medida:

```
=makedate(transaction_year,transaction_month)
```

Tabela de resultados

transaction_year	transaction_month	transaction_date
2018	08	08/01/2018
2018	09	09/01/2018

A medida `transaction_date` é criada no objeto de gráfico usando a função `makedate()` e informando os campos de ano e mês como argumentos da função.

A função então combina esses valores, bem como o valor do dia assumido de 01. Esses valores são então convertidos em um campo de data, retornando os resultados no formato da variável de sistema `DateFormat`.

Exemplo 4: Cenário

Script de carregamento e expressão de gráfico

Visão geral

Crie um conjunto de dados do calendário para o ano civil de 2022.

Script de carregamento

```
SET DateFormat='MM/DD/YYYY';

Calendar:
       load
                *
    where year(date)=2022;
load
       date(recno()+makedate(2021,12,31)) as date
AutoGenerate 400;
```

Resultados

Tabela de resultados

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

date
01/22/2022
01/23/2022
01/24/2022
01/25/2022
Mais de 340 linhas

A função `makedate()` cria um valor de data para 31 de dezembro de 2021. A função `recno()` fornece o número do registro atual que está sendo carregado na tabela, começando em 1. Portanto, o primeiro registro tem a data de 1º de janeiro de 2022. Cada `recno()` sucessiva então incrementará essa data em 1. Essa expressão é agrupada em uma função `date()` para converter o valor em uma data. Esse processo é repetido 400 vezes pela função `autogenerate`. Finalmente, usando um carregamento anterior, uma condição `where` pode ser usada para carregar somente datas do ano de 2022. Esse script gera um calendário contendo todas as datas de 2022.

maketime

Esta função retorna um momento calculado a partir da hora **hh**, do minuto **mm** e do segundo **ss**.

Sintaxe:

```
MakeTime(hh [ , mm [ , ss ] ])
```

Tipo de dados de retorno: dual

Argumentos

Argumento	Descrição
hh	A hora como um inteiro.
mm	O minuto como um inteiro. Se nenhum minuto for indicado, 00 será assumido.
ss	O segundo como um inteiro. Se nenhum segundo for indicado, 00 será assumido.

Quando usar

A função `maketime()` normalmente seria usada no script para geração de dados para gerar um campo de hora. Às vezes, quando o campo de hora é derivado do texto de entrada, essa função pode ser usada para construir a hora usando seus componentes.

Estes exemplos usam o formato de hora `h:mm:ss`. O formato de hora é especificado no comando `SET TimeFormat` na parte superior do seu script de carregamento de dados. Altere o formato nos exemplos para atender às suas necessidades.

Exemplos de funções

Exemplo	Resultado
`maketime(22)`	Retorna 22:00:00.
`maketime(22, 17)`	Retorna 22:17:00.
`maketime(22,17,52 )`	Retorna 22:17:52.

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplo 1: maketime()

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações que é carregado em uma tabela denominada `Transactions`
- Os tempos de transação são fornecidos em três campos: `hours`, `minutes` e `seconds`.
- A criação de um campo, `transaction_time`, que retorna a hora no formato da `variável` de sistema TimeFormat.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `transaction_hour`
- `transaction_minute`
- `transaction_second`
- `transaction_time`

Tabela de resultados

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

O campo `transaction_time` é criado na instrução de carregamento anterior usando a função `maketime()` e informando os campos de hora, minuto e segundo como argumentos da função.

Em seguida, a função combina e converte esses valores em um campo de tempo, retornando os resultados no formato de tempo da variável de sistema `TimeFormat`.

Exemplo 2: função time()

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- O mesmo conjunto de dados e cenário do primeiro exemplo.
- A criação de um campo, `transaction_time`, que nos permitirá mostrar os resultados no formato de 24 horas sem modificar a variável de sistema `TimeFormat`.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `transaction_hour`
- `transaction_minute`
- `transaction_second`
- `transaction_time`

Tabela de resultados

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

Nesse caso, a função `maketime()` está aninhada dentro da função `time()`. O segundo argumento da função `time()` define o formato dos resultados da função `maketime()` como a `h:mm:ss` necessária.

Exemplo 3: Exemplo de objeto de gráfico

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações que é carregado em uma tabela denominada `Transactions`.
- Os tempos de transação são fornecidos em dois campos: `hours` e `minutes`.
- A criação de um campo, `transaction_time`, que retorna a hora no formato da variável de sistema `TimeFormat`.

Crie uma medida de objeto de gráfico, `transaction_time`, que retorna uma hora no formato `h:mm:ss TT`.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `transaction_hour`
- `transaction_minute`

Para calcular a `transaction_time`, crie esta medida:

```
=maketime(transaction_hour,transaction_minute)
```

Tabela de resultados

transaction_hour	transaction_minute	=maketime(transaction_hour, transaction_minute)
2	52	2:52:00 AM

transaction_hour	transaction_minute	=maketime(transaction_hour, transaction_minute)
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

A medida `transaction_time` é criada no objeto de gráfico usando a função `maketime()` e informando os campos de hora e minuto como argumentos da função.

A função então combina esses valores, e os segundos são considerados 00. Esses valores são então convertidos em um campo de hora, retornando os resultados no formato da variável de sistema `TimeFormat`.

Exemplo 4: Cenário

Script de carregamento e expressão de gráfico

Visão geral

Crie um conjunto de dados do calendário para o mês de janeiro de 2022, dividido em incrementos de oito horas.

Script de carregamento

```
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;
```

Resultados

Tabela de resultados

carimbo de data/hora
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
Mais de 68 linhas

A função `autogenerate` inicial cria um calendário contendo todas as datas de janeiro em uma tabela denominada `tmpCalendar`.

Uma segunda tabela, contendo três registros, é criada. Para cada registro, `recno() - 1` é obtido (valores 0, 1, 2), e o resultado é multiplicado por 8. Como resultado, isso gera os valores 0, 8, 16. Esses valores são usados como o parâmetro de hora em uma função `maketime()`, com valores de minuto e segundo de 0. Como resultado, a tabela contém três campos de horário: 12:00:00 AM, 8:00:00 AM e 4:00:00 PM.

Essa tabela está unida à tabela `tmpCalendar`. Como não há campos correspondentes entre as duas tabelas para a união, as linhas de tempo são adicionadas a cada linha de data. Como resultado, cada linha de data agora é repetida três vezes com cada valor de tempo.

Finalmente, a tabela de calendário é criada a partir de um carregamento residente da tabela `tmpCalendar`. Os campos de data e hora são concatenados e agrupados na função `timestamp()` para criar o campo de carimbo de data/hora.

A tabela `tmpCalendar` é então descartada.

makeweekdate

Essa função retorna uma data calculada a partir do ano, do número da semana e do dia da semana.

Sintaxe:

```
MakeWeekDate(weekyear [, week [, weekday [, first_week_day [, broken_weeks [, reference_day]]]]])
```

Tipo de dados de retorno: dual

A função `makeweekdate()` está disponível como função de script e gráfico. Ela calculará a data com base nos parâmetros informados para a função.

Argumentos

Argumento	Descrição
weekyear	O ano definido pela função `weekYear()` para a data específica, ou seja, o ano ao qual o número da semana pertence. *Em alguns casos, o ano da semana pode ser diferente do ano civil, por exemplo, se a semana 1 começar já em dezembro do ano anterior.*
week	O número da semana, conforme definido pela função `week()` para a data específica. Se nenhum número da semana for declarado, assume-se 1.

Argumento	Descrição
weekday	O dia da semana definido pela função `WeekDay()` para a data em questão. 0 é o primeiro dia da semana e 6 é o último dia da semana. Se nenhum dia da semana for informado, assume-se 0. *Embora 0 sempre signifique o primeiro dia da semana e 6 seja sempre o último, os dias da semana a que correspondem são determinados pelo parâmetro **first_week_day**. Se omitido, o valor da variável **FirstWeekDay** será usado.* Se semanas interrompidas forem usadas, juntamente com uma combinação impossível de parâmetros, isso poderá levar a um resultado que não pertence ao ano escolhido. **Exemplo:** `MakeWeekDate(2021,1,0,6,1)` Retorna '27 de dezembro de 2020', pois esse dia é o primeiro dia (o domingo) da semana especificada. 1º de janeiro de 2021 foi uma sexta-feira.
first_week_day	Especifica o dia no qual inicia a semana. Se omitido, o valor da variável **FirstWeekDay** é usado. Os valores possíveis **first_week_day** são 0 para segunda-feira, 1 para terça, 2 para quarta-feira, 3 para quinta-feira, 4 para sexta-feira, 5 para sábado e 6 para domingo. Para obter mais informações sobre a variável de sistema, consulte *FirstWeekDay (page 226)*.
broken_weeks	Se você não especificar **broken_weeks**, o valor da variável **BrokenWeeks** será usado para definir se as semanas estão interrompidas ou não.
reference_day	Se você não especificar **reference_day**, o valor da variável **ReferenceDay** será usado para definir qual dia de janeiro definir como dia de referência para definir a semana 1.

Quando usar

A função `makeweekdate()` normalmente seria usada no script para geração de dados, para gerar uma lista de datas ou criar datas em que o ano, a semana e o dia da semana são fornecidos nos dados de entrada.

Os exemplos a seguir pressupõem que:

```
SET FirstWeekDay=0;
SET BrokenWeeks=0;
SET ReferenceDay=4;
```

Exemplos de funções

Exemplo	Resultado
`makeweekdate(2014,6,6)`	retorna 02/09/2014
`makeweekdate(2014,6,1)`	retorna 02/04/2014
`makeweekdate(2014,6)`	retorna 02/03/2014 (dia da semana 0 é exibido)

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplo 1: dia incluído

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo o total de vendas semanais para 2022 em uma tabela denominada `Sales`.
- As datas das transações são fornecidas em três campos: `year`, `week` e `sales`.
- Um carregamento anterior, usada para criar uma medida `end_of_week`, usando a função `makeweekdate()` para retornar a data da sexta-feira dessa semana no formato MM/DD/AAAA.

Para provar que a data retornada é uma sexta-feira, a expressão `end_of_week` também é incluída na função `weekday()` para mostrar o dia da semana.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `transaction_year`
- `transaction_week`
- `end_of_week`
- `week_day`

Tabela de resultados

transaction_year	transaction_week	end_of_week	week_day
2022	01	01/07/2022	Fri
2022	02	01/14/2022	Fri
2022	03	01/21/2022	Fri
2022	04	01/28/2022	Fri
2022	05	02/04/2022	Fri
2022	06	02/11/2022	Fri
2022	07	02/18/2022	Fri

O campo `end_of_week` é criado na instrução de carregamento anterior usando a função `makeweekdate()`. Os campos `transaction_year`, `transaction_week` são informados pela função como argumentos de ano e semana. Um valor de 4 é usado para o argumento “day”.

A função então combina e converte esses valores em um campo de data, retornando os resultados no formato da variável de sistema `DateFormat`.

A função `makeweekdate()` e seus argumentos também são agrupados em uma função `weekday()` para retornar o campo `week_day`; e, como pode ser visto na tabela acima, o campo `week_day` mostra que essas datas ocorrem em uma sexta-feira.

Exemplo 2: dia excluído

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo os totais de vendas semanais para 2022 em uma tabela denominada `Sales`.
- As datas das transações são fornecidas em três campos: `year`, `week` e `sales`.
- Um carregamento anterior, usado para criar uma medida, `first_day_of_week`, usando a função `makeweekdate()`. Isso retornará a data de segunda-feira dessa semana no formato MM/DD/AAAA.

Para provar que a data retornada é uma segunda-feira, a expressão `first_day_of_week` também é incluída na função `weekday()` para mostrar o dia da semana.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `transaction_year`
- `transaction_week`

- first_day_of_week
- week_day

Tabela de resultados

transaction_year	transaction_week	first_day_of_week	week_day
2022	01	01/03/2022	Mon
2022	02	01/10/2022	Mon
2022	03	01/17/2022	Mon
2022	04	01/24/2022	Mon
2022	05	01/31/2022	Mon
2022	06	02/07/2022	Mon
2022	07	02/14/2022	Mon

O campo first_day_of_week é criado na instrução de carregamento anterior usando a função makeweekdate(). Os parâmetros transaction_year e transaction_week são informados como argumentos de função, e o parâmetro “day” é deixado em branco.

A função então combina e converte esses valores em um campo de data, retornando os resultados no formato da variável de sistema DateFormat.

A função makeweekdate() e seus argumentos também são agrupados em uma função weekday() para retornar o campo week_day. Como pode ser visto na tabela acima, o campo week_day retorna segunda-feira em todos os casos desde que esse parâmetro foi deixado em branco na função makeweekdate(), cujo padrão é 0 (primeiro dia da semana) e o primeiro dia da semana é definido como segunda-feira pela variável do sistema FirstWeekDay.

Exemplo 3: Exemplo de objeto de gráfico

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo os totais de vendas semanais para 2022 em uma tabela denominada Sales.
- As datas das transações são fornecidas em três campos: year, week e sales.

Neste exemplo, um objeto de gráfico será usado para criar uma medida equivalente ao cálculo de end_of_week do primeiro exemplo. Essa medida usará a função makeweekdate() para retornar a data da sexta-feira daquela semana no formato MM/DD/AAAA.

Para provar que a data retornada é uma sexta-feira, uma segunda medida é criada para retornar o dia da semana.

Script de carregamento

```
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
];
```

Resultados

Faça o seguinte:

1. Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:
   - `transaction_year`
   - `transaction_week`
2. Para realizar o cálculo equivalente ao do campo `end_of_week` do primeiro exemplo, crie a seguinte medida:
   `=makeweekdate(transaction_year,transaction_week,4)`
3. Para calcular o dia da semana para cada transação, crie a seguinte medida:
   `=weekday(makeweekdate(transaction_year,transaction_week,4))`

Tabela de resultados

transaction_ year	transaction_ week	=makeweekdate (transaction_ year,transaction_week,4)	=weekday(makeweekdate (transaction_year,transaction_ week,4))
2022	01	01/07/2022	Fri
2022	02	01/14/2022	Fri
2022	03	01/21/2022	Fri
2022	04	01/28/2022	Fri
2022	05	02/04/2022	Fri
2022	06	02/11/2022	Fri
2022	07	02/18/2022	Fri

Um campo equivalente a `end_of_week` é criado no objeto de gráfico como uma medida usando a função `makeweekdate()`. Os campos `transaction_year` e `transaction_week` são transmitidos como argumentos "year" e "week". Um valor de 4 é usado para o argumento "day".

A função então combina e converte esses valores em um campo de data, retornando os resultados no formato da variável de sistema `DateFormat`.

A função `makeweekdate()` e seus argumentos também são agrupados em uma função `weekday()` para retornar um cálculo equivalente ao do campo `week_day` do primeiro exemplo. Como pode ser visto na tabela acima, a última coluna à direita mostra que essas datas ocorrem em uma sexta-feira.

Exemplo 4: Cenário

Script de carregamento e expressão de gráfico

Visão geral

Neste exemplo, crie uma lista de datas contendo todas as sextas-feiras do ano de 2022.

Abra o editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia.

Script de carregamento

```
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;
```

Resultados

Tabela de resultados

date	weekday
01/07/2022	Fri
01/14/2022	Fri
01/21/2022	Fri
01/28/2022	Fri
02/04/2022	Fri

date	weekday
02/11/2022	Fri
02/18/2022	Fri
02/25/2022	Fri
03/04/2022	Fri
03/11/2022	Fri
03/18/2022	Fri
03/25/2022	Fri
04/01/2022	Fri
04/08/2022	Fri
04/15/2022	Fri
04/22/2022	Fri
04/29/2022	Fri
05/06/2022	Fri
05/13/2022	Fri
05/20/2022	Fri
05/27/2022	Fri
06/03/2022	Fri
06/10/2022	Fri
06/17/2022	Fri
Mais de 27 linhas	

A função `makeweekdate()` localiza todas as sextas-feiras em 2022. Usar um parâmetro “week” de -2 garante que nenhuma data seja perdida. Finalmente, um carregamento anterior cria um campo `weekday` adicional para maior clareza, para mostrar que cada valor de `date` é uma sexta-feira.

minute

Esta função retorna um número inteiro que representa o minuto em que a fração da **expression** é interpretada como uma hora, de acordo com a interpretação numérica padrão.

Sintaxe:

```
minute(expression)
```

Tipo de dados de retorno: inteiro

Quando usar

A função `minute()` é útil quando você deseja comparar agregações por minuto. Por exemplo, você pode usar essa função para ver a distribuição da contagem de atividades por minuto.

Essas dimensões podem ser criadas no script de carregamento usando a função para criar um campo em uma tabela de Calendário mestre. Como alternativa, elas podem ser usadas diretamente em um gráfico como uma dimensão calculada.

Exemplos de funções

Exemplo	Resultado
`minute ( '09:14:36' )`	Retorna 14.
`minute ( '0.5555' )`	Retorna 19 (porque 0.5555 = 13:19:55).

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplo 1 – Variável (script)

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo transações por carimbo de data/hora que é carregado em uma tabela denominada `Transactions`.
- A `variável` de sistema TimeStamp padrão (`M/D/YYYY h:mm:ss[.fff] TT`) é usada.
- A criação de um campo, `minute`, para calcular quando as transações ocorreram.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `timestamp`
- `minute`

Tabela de resultados

carimbo de data/hora	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

Os valores no campo `minute` são criados usando a função `minute()` e informando `timestamp` como a expressão na instrução de carregamento anterior.

Exemplo 2 – Objeto de gráfico (gráfico)

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- O mesmo conjunto de dados e cenário do primeiro exemplo.
- A variável de sistema `TimeStamp` padrão (`M/D/YYYY h:mm:ss[.fff] TT`) é usada.

No entanto, neste exemplo, o conjunto de dados inalterado é carregado no aplicativo. Os valores de `minute` são calculados por meio de uma medida em um objeto de gráfico.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão: `timestamp`.

Crie a seguinte medida:

```
=minute(timestamp)
```

Tabela de resultados

carimbo de data/hora	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

Os valores para `minute` são criados usando a função `minute()` e informando `timestamp` como a expressão em uma medida para o objeto de gráfico.

Exemplo 3: Cenário

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados de carimbos de data/hora, que é gerado para representar entradas em uma barreira de tíquete.
- Informações com cada `timestamp` e suas `id` correspondentes, que são carregadas em uma tabela denominada `Ticket_Barrier_Tracker`.
- A variável de sistema `TimeStamp` padrão (`M/D/YYYY h:mm:ss[.fff] TT`) é usada.

O usuário gostaria de um objeto de gráfico que mostrasse, por minuto, a contagem de entradas de barreira.

Script de carregamento

```
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;
```

Resultados

Faça o seguinte:

1. Carregue os dados e abra uma pasta. Crie uma nova tabela.
2. Crie uma dimensão calculada usando a seguinte expressão:
   `=minute(timestamp)`
3. Adicione a seguinte medida de agregação para calcular a contagem total de entradas:
   `=count(id)`
4. Defina o **Formato numérico** da medida como **Dinheiro**.

Tabela de resultados

minute(timestamp)	=count(id)
0	174
1	171
2	175
3	165
4	188
5	176
6	158
7	187
8	178
9	178
10	197
11	161
12	166
13	184

minute(timestamp)	=count(id)
14	159
15	161
16	152
17	160
18	176
19	164
20	170
21	170
22	142
23	145
24	155
Mais de 35 linhas	

month

Esta função retorna um valor dual: um nome do mês, conforme definido na variável de ambiente **MonthNames** e um inteiro entre 1 e 12. O mês é calculado a partir da data de interpretação da expressão, de acordo com a interpretação do número padrão.

A função retorna o nome do mês no formato da `variável` do sistema MonthName para uma data específica. Ela é geralmente usada para criar um campo de dia como uma dimensão em um Calendário mestre.

Sintaxe:

```
month(expression)
```

Tipo de dados de retorno: inteiro

Exemplos de funções

Exemplo	Resultado
`month( 2012-10-12 )`	retorna Oct
`month( 35648 )`	retorna Aug, por que 35648 = 1997-08-06

Exemplo 1 – conjunto de dados DateFormat (script)

Script de carregamento e resultados

Visão geral

Abra o Data load editor e adicione o script de carregamento abaixo em uma nova guia.Editor de carregamento de dados

O script de carregamento contém:

- Um conjunto de dados de datas chamado Master_Calendar. A variável do sistema `DateFormat` é definida como DD/MM/AAAA.
- Um carregamento anterior que cria um campo adicional, chamado`month_name`, usando a função `month()`.
- Um campo adicional, chamado `long_date`, usando a função `date()` para expressar a data completa.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `long_date`
- `month_name`

Tabela de resultados

data	long_date	month_name
03/01/2022	03-Janeiro- 2022	Jan
03/02/2022	03-Fevereiro- 2022	Fev
03/03/2022	03-Março- 2022	Mar
03/04/2022	03-Abril- 2022	Abr
03/05/2022	03-Maio- 2022	Maio
03/06/2022	03-Junho- 2022	Jun
03/07/2022	03-Julho- 2022	Jul
03/08/2022	03-Agosto- 2022	Ago
03/09/2022	03-Setembro- 2022	Set
03/10/2022	03-Outubro- 2022	Out
03/11/2022	03-Novembro- 2022	Nov

O nome do mês é avaliado corretamente pela função `month()` no script.

Exemplo 2 – datas ANSI (script)

Script de carregamento e resultados

Visão geral

Abra o Editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados de datas chamado `Master_Calendar`. A variável de sistema DateFormat DD/MM/AAAA é usada. No entanto, as datas incluídas no conjunto de dados estão no formato de data padrão ANSI.
- Um carregamento anterior que cria um campo adicional, chamado `month_name`, usando a função month().
- Um campo adicional, chamado `long_date`, usando a função para expressar a data completa.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `long_date`
- `month_name`

Tabela de resultados

data	long_date	month_name
03/11/2022	11-Março- 2022	11
03/12/2022	12-Março- 2022	12
03/13/2022	13-Março- 2022	13
03/14/2022	14-Março- 2022	14
03/15/2022	15-Março- 2022	15
03/16/2022	16-Março- 2022	16
03/17/2022	17-Março- 2022	17
03/18/2022	18-Março- 2022	18
03/19/2022	19-Março- 2022	19
03/20/2022	20-Março- 2022	20
03/21/2022	21-Março- 2022	21

O nome do mês é avaliado corretamente pela função `month()` no script.

Exemplo 3 – Datas não formatadas (script)

Script de carregamento e resultados

Visão geral

Abra o Editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados de datas chamado `Master_Calendar`. A variável de sistema `DateFormat` DD/MM/AAAA é usada.
- Um carregamento anterior que cria um campo adicional, chamado `month_name`, usando a função `month()`.
- A data original não formatada, chamada `unformatted_date`.
- Um campo adicional, chamado `long_date`, usando a função `date()` para expressar a data completa.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `unformatted_date`
- `long_date`
- `month_name`

Tabela de resultados

unformatted_date	long_date	month_name
44868	03-Janeiro- 2022	Jan
44898	03-Fevereiro- 2022	Fev
44928	03-Março- 2022	Mar
44958	03-Abril- 2022	Abr
44988	03-Maio- 2022	Maio
45018	03-Junho- 2022	Jun
45048	03-Julho- 2022	Jul
45078	03-Agosto- 2022	Ago
45008	03-Setembro- 2022	Set
45038	03-Outubro- 2022	Out
45068	03-Novembro- 2022	Nov

O nome do mês é avaliado corretamente pela função `month()` no script.

Exemplo 4 – Calculando o mês de vencimento

Script de carregamento e expressão de gráfico

Visão geral

Abra o Editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados de pedidos feitos em março chamado `Subscriptions`. A tabela contém três campos:
    - id
    - order_date
    - amount

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão: `order_date`.

Para calcular o mês de expiração de um pedido, crie esta medida `=month(order_date+180)`.

Tabela de resultados

order_date	=month(order_date+180)
03/01/2022	Jul
03/02/2022	Ago
03/03/2022	Ago
03/04/2022	Set
03/05/2022	Out
03/06/2022	Nov
03/07/2022	Dez
03/08/2022	Jan
03/09/2022	Mar
03/10/2022	Abr
03/11/2022	Maio

A função `month()` determina corretamente que um pedido feito em 11 de março expiraria em julho.

monthend

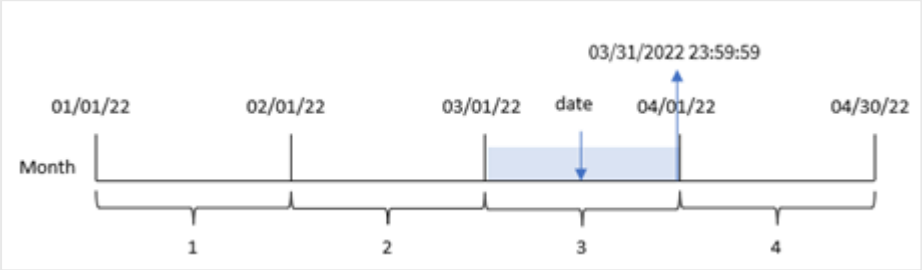
Esta função retorna um valor correspondente ao carimbo de data/hora do último milissegundo do último dia do mês que contém `date`. O formato de saída padrão será o `DateFormat` definido no script.

Sintaxe:

```
MonthEnd(date[, period_no])
```

Em outras palavras, a função `monthend()` determina em qual mês a data cai. Em seguida, ela retorna um carimbo de data/hora, em formato de data, para o último milissegundo daquele mês.

Diagrama da função `monthend`.



Quando usar

A função `monthend()` é usada como parte de uma expressão quando você deseja que o cálculo use a fração do mês que ainda não ocorreu. Por exemplo, se você quiser calcular o total de juros ainda não acumulados durante o mês.

Tipo de dados de retorno: dual

Argumentos

Argumento	Descrição
date	A data ou o carimbo de data/hora a ser avaliado.
period_no	**period_no** é um inteiro, no qual, se 0 ou omitido, indica o mês que contém **date**. Valores negativos em **period_no** indicam meses precedentes e valores positivos indicam meses sucessivos.

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplos de funções

Exemplo	Resultado
`monthend('02/19/2012')`	Retorna 02/29/2012 23:59:59.
`monthend('02/19/2001', -1)`	Retorna 01/31/2001 23:59:59.

Exemplo 1: Exemplo básico

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações para 2022, que é carregado em uma tabela denominada "Transactions".
- Um campo de data na `variável` de sistema DateFormat, no formato (`MM/DD/YYYY`).
- Um comando de carregamento anterior que contém:
  - A função `monthend()` que é definida como o campo “`end_of_month`”.
  - A função `timestamp` que é definida como o campo “`end_of_month_timestamp`”.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- id
- date
- end_of_month
- end_of_month_timestamp

Tabela de resultados

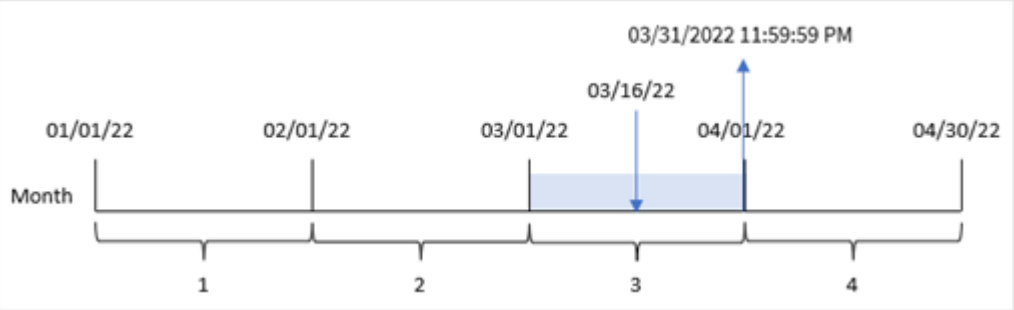
id	date	end_of_month	end_of_month_timestamp
8188	1/7/2022	01/31/2022	1/31/2022 11:59:59 PM
8189	1/19/2022	01/31/2022	1/31/2022 11:59:59 PM
8190	2/5/2022	02/28/2022	2/28/2022 11:59:59 PM
8191	2/28/2022	02/28/2022	2/28/2022 11:59:59 PM
8192	3/16/2022	03/31/2022	3/31/2022 11:59:59 PM
8193	4/1/2022	04/30/2022	4/30/2022 11:59:59 PM
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

id	date	end_of_month	end_of_month_timestamp
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

O campo "end_of_month" é criado na instrução de carregamento anterior usando a função `monthend()` e transmitindo o campo de data como o argumento da função.

A função `monthend()` identifica em qual mês o valor da data cai, retornando um carimbo de data/hora para o último milissegundo desse mês.

Diagrama da função `monthend` com março como o mês selecionado.



A transação 8192 ocorreu em 16 de março. A função `monthend()` retorna o último milissegundo desse mês, que é 31 de março às 11:59:59 PM.

Exemplo 2: period_no

Script de carregamento e resultados

Visão geral

São usados o mesmo conjunto de dados e cenário do primeiro exemplo.

Neste exemplo, a tarefa é criar um campo, “`previous_month_end`”, que retorna o carimbo de data/hora do final do mês antes da transação.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `id`
- `date`
- `previous_month_end`
- `previous_month_end_timestamp`

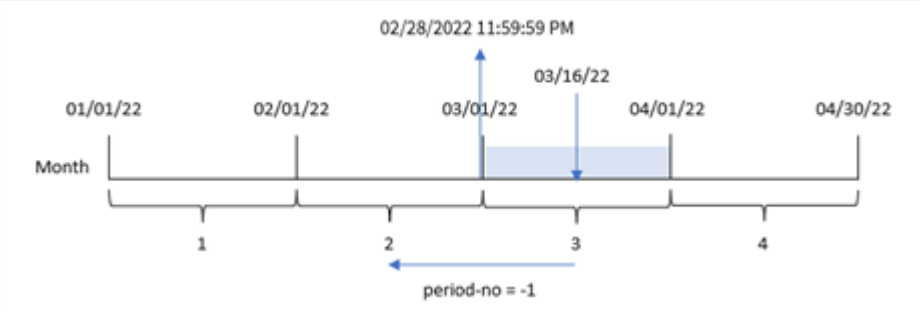
Tabela de resultados

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

id	date	previous_month_end	previous_month_end_timestamp
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

A função `monthend()`primeiro identifica o mês em que as transações ocorrem como um `period_no` de `-1` usado como argumento offset. Em seguida, ela muda para um mês antes e identifica o último milissegundo desse mês.

Diagrama da função `monthend` com a variável period_no.



A transação 8192 ocorreu em 16 de março. A função `monthend()` identifica que o mês anterior à realização da transação foi em fevereiro. Em seguida, ela retorna o último milissegundo desse mês, 28 de fevereiro às 11:59:59 PM.

Exemplo 3: Exemplo de gráfico

Script de carregamento e expressão de gráfico

Visão geral

São usados o mesmo conjunto de dados e cenário do primeiro exemplo.

Neste exemplo, o conjunto de dados permanece inalterado e carregado no aplicativo. A tarefa é criar um cálculo que retorne um timestamp do final do mês em que ocorreram as transações como medida em um gráfico do aplicativo.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `id`

Para calcular a data de término do mês em que uma transação ocorre, crie as seguintes medidas:

- `=monthend(date)`
- `=timestamp(monthend(date))`

Tabela de resultados

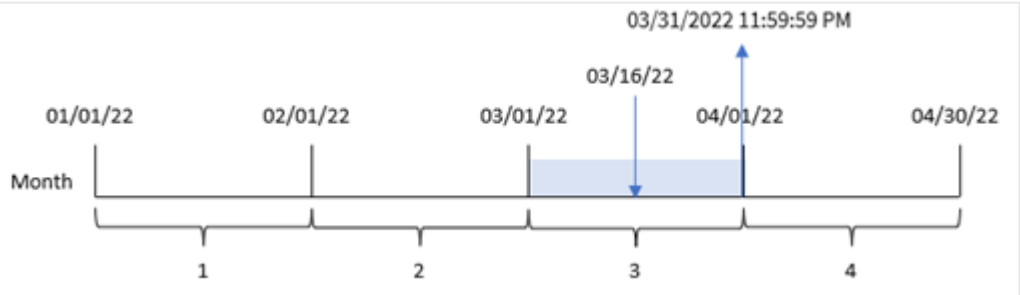
id	date	=monthend(date)	=timestamp(monthend(date))
8188	10/14/2022	10/31/2022	10/31/2022 11:59:59 PM

id	date	=monthend(date)	=timestamp(monthend(date))
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
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

A medida "end_of_month" é criada no objeto de gráfico usando a função `monthend()` e transmitindo o campo de data como o argumento da função.

A função `monthend()` identifica em qual mês o valor da data cai e retorna um carimbo de data/hora para o último milissegundo desse mês.

Diagrama da função `monthend` com a variável period_no.



A transação 8192 ocorreu em 16 de março. A função `monthend()` retorna o último milissegundo desse mês, que é 31 de março às 11:59:59 PM.

Exemplo 4: Cenário

Script de carregamento e resultados

Visão geral

Neste exemplo, um conjunto de dados é carregado em uma tabela chamada "`Employee_Expenses`". A tabela contém os seguintes campos:

- IDs de funcionários
- Nomes de funcionários
- As reivindicações de despesas médias diárias de cada funcionário.

O usuário final deseja um gráfico que mostra, por ID de funcionário e nome do funcionário, a declaração de despesas estimada para o restante do mês.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `employee_id`
- `employee_name`

Para calcular os juros acumulados, crie esta medida:

```
=floor(monthend(today(1),0)-today(1))*avg_daily_claim
```

Essa medida é dinâmica e produzirá resultados de tabela diferentes, dependendo da data em que você carrega os dados.

Defina o **Formato numérico** da medida como **Dinheiro**.

Tabela de resultados

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

A função `monthend()` retorna a data de término do mês atual usando a data de hoje como seu único argumento. A expressão retorna o número de dias que restam neste mês, subtraindo a data de hoje da data de término do mês.

Esse valor é então multiplicado pela média de solicitações de despesas diárias por cada funcionário para calcular o valor estimado das solicitações que cada funcionário deve fazer no mês restante.

monthname

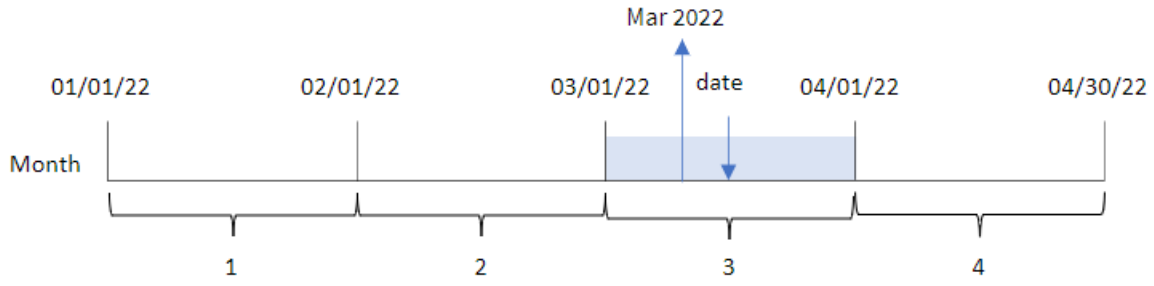
Esta função retorna um valor de exibição que mostra o mês (formatado de acordo com a variável de script **MonthNames**) e o ano com um valor numérico subjacente que corresponde a um carimbo de hora do primeiro milissegundo do primeiro dia do mês.

Sintaxe:

```
MonthName(date[, period_no])
```

Tipo de dados de retorno: dual

Diagrama da função monthname



Argumentos

Argumento	Descrição
date	A data ou o carimbo de data/hora a ser avaliado.
period_no	**period_no** é um inteiro, no qual, se 0 ou omitido, indica o mês que contém **date**. Valores negativos em **period_no** indicam meses precedentes e valores positivos indicam meses sucessivos.

Exemplos de funções

Exemplo	Resultado
`monthname('10/19/2013')`	Retorna Oct 2013
`monthname('10/19/2013', -1)`	Retorna Sep 2013

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplo 1: Exemplo básico

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações para 2022, que é carregado em uma tabela denominada `Transactions`.
- O campo de data fornecido no formato da `variável` de sistema DateFormat (MM/DD/AAAA).
- A criação de um campo, `transaction_month`, que retorna o mês em que as transações ocorreram.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `transaction_month`

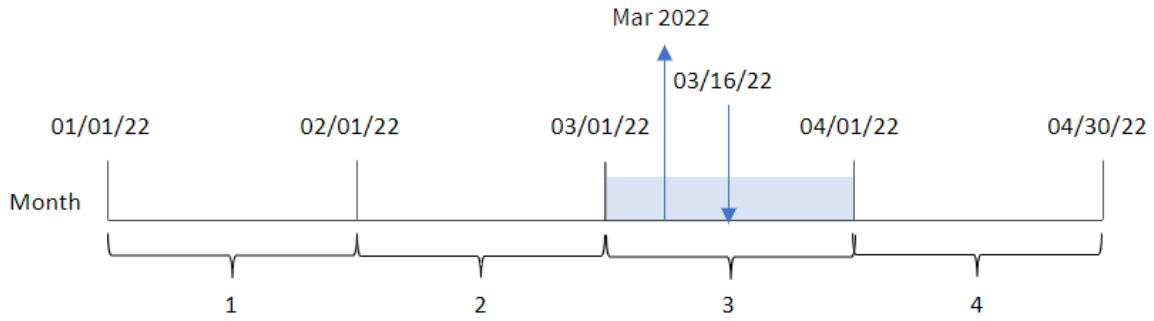
Tabela de resultados

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

date	transaction_month
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

O campo `transaction_month` é criado na instrução de carregamento anterior usando a função `monthname()` e transmitindo o campo `date` como o argumento da função.

Diagrama da função monthname, exemplo básico



A função `monthname()` identifica que a transação 8192 ocorreu em março de 2022 e retorna esse valor usando a variável do sistema `MonthNames`.

Exemplo 2: period_no

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- O mesmo conjunto de dados inline e cenário do primeiro exemplo.
- A criação de um campo, `transaction_previous_month`, que retorna o carimbo de data/hora do final do mês anterior à transação.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

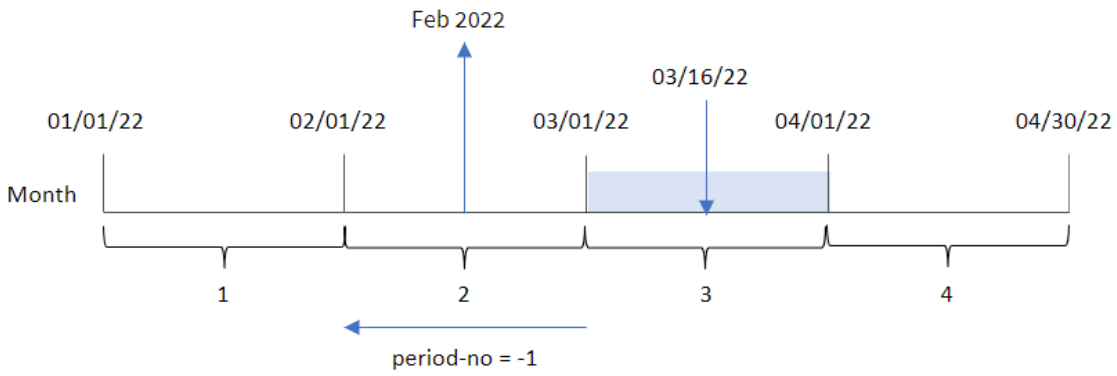
- `date`
- `transaction_previous_month`

Tabela de resultados

date	transaction_previous_month
1/7/2022	Dez 2021
1/19/2022	Dez 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
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

Nesse caso, como um `period_no` de `-1` foi usado como o argumento de deslocamento na função `monthname()`, a função primeiro identifica o mês em que as transações ocorrem. Em seguida, ela muda para um mês anterior e retorna o nome desse mês e o ano.

Diagrama da função monthname, exemplo de period_no



A transação 8192 ocorreu em 16 de março. A função `monthname()` identifica que o mês anterior à transação foi fevereiro e retorna esse mês, no formato da variável do sistema `MonthNames`, junto com o ano 2022.

Exemplo 3: Exemplo de objeto de gráfico

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém o mesmo conjunto de dados inline e cenário do primeiro exemplo. No entanto, neste exemplo, o conjunto de dados inalterado é carregado no aplicativo. O cálculo que retorna um carimbo de data/hora para o final do mês em que as transações ocorreram é criado como uma medida em um objeto de gráfico do aplicativo.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão:`date`.

Crie a seguinte medida:

```
=monthname(date)
```

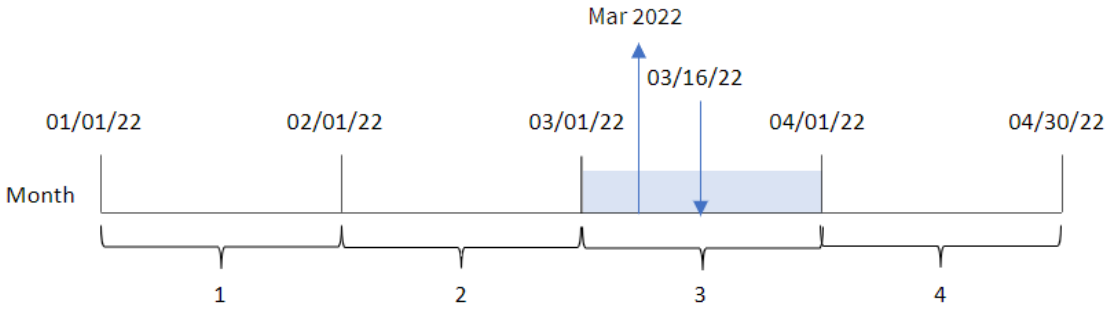
Tabela de resultados

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
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

date	=monthname(date)
9/26/2022	Sep 2022
10/14/2022	Oct 2022
10/29/2022	Oct 2022

A medida `month_name` é criada no objeto de gráfico usando a função `monthname()` e transmitindo o campo `date` como argumento da função.

Diagrama da função monthname, exemplo de objeto de gráfico



A função `monthname()` identifica que a transação 8192 ocorreu em março de 2022 e retorna esse valor usando a variável do sistema `MonthNames`.

monthsend

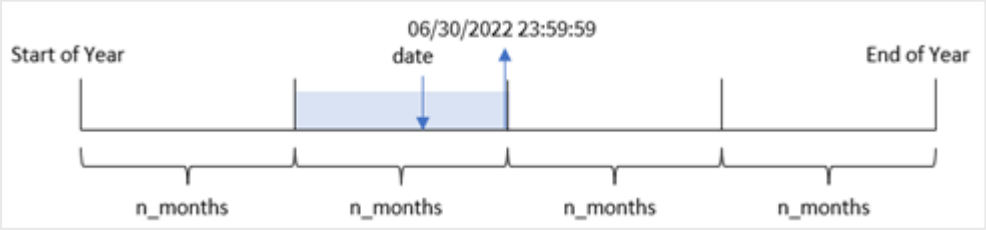
Essa função retorna um valor correspondente ao carimbo de data/hora do último milissegundo do mês, bimestre, trimestre, quadrimestre ou semestre contendo uma data base. Também é possível encontrar o carimbo de data/hora para o final de um período anterior ou seguinte. O formato de saída padrão é o `DateFormat` definido no script.

Sintaxe:

```
MonthsEnd(n_months, date[, period_no [, first_month_of_year]])
```

Tipo de dados de retorno: dual

Diagrama da função monthsend.



Argumentos

Argumento	Descrição
n_months	O número de meses que define o período. Um inteiro ou uma expressão que se resolve como um inteiro que deve ser um dos seguintes: 1 (equivalente à função inmonth()), 2 (bimestre), 3 (equivalente à função inquarter()), 4 (quadrimestre) ou 6 (semestre).
date	A data ou o carimbo de data/hora a ser avaliado.
period_no	O período pode ser deslocado por **period_no**, um inteiro ou expressão que resolve um inteiro, no qual o valor 0 indica o período que contém **base_date**. Valores negativos em **period_no** indicam períodos precedentes e valores positivos indicam períodos sucessivos.
first_month_of_year	Se desejar trabalhar com anos (fiscais) que não comecem em janeiro, indique um valor entre 2 e 12 em **first_month_of_year**.

A função `monthsend()` divide o ano em segmentos com base no argumento `n_months` fornecido. Em seguida, ele avalia em qual segmento cada data fornecida se encaixa e retorna o último milissegundo, em formato de data, desse segmento. A função pode retornar o carimbo de data/hora final dos segmentos anteriores ou seguintes, bem como redefinir o primeiro mês do ano.

Os seguintes segmentos do ano estão disponíveis na função como argumentos `n_month`.

Argumentos `n_month`

Período	Número de meses
mês	1
bimestre	2
trimestre	3
quadrimestre	4
semestre	6

Quando usar

A função `monthsend()` é usada como parte de uma expressão quando o usuário deseja que o cálculo use a fração do mês decorrido até o momento. O usuário tem a oportunidade, usando uma variável, de selecionar o período de sua escolha. Por exemplo, `monthsend()` pode fornecer uma variável de entrada para permitir que o usuário calcule o total de juros ainda não acumulados durante o mês, trimestre ou semestre.

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplos de funções

Exemplo	Resultado
`monthsend(4, '07/19/2013')`	Retorna 08/31/2013.
`monthsend(4, '10/19/2013', -1)`	Retorna 08/31/2013.
`monthsend(4, '10/19/2013', 0, 2)`	Retorna 01/31/2014. Porque o início do ano passa a ser o mês 2.

Exemplo 1: Exemplo básico

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações para 2022 é carregado em uma tabela denominada "`Transactions`".
- Um campo de data fornecido no formato da variável de sistema `DateFormat (MM/DD/YYYY)`.
- Uma instrução de carregamento anterior contendo:
    - A função `monthsend` que é definida como o campo, “`bi_monthly_end`”. Isso agrupa as transações em segmentos bimestrais.
    - A função `timestamp` que retorna o carimbo de data/hora inicial do segmento para cada transação.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `id`
- `date`
- `bi_monthly_end`
- `bi_monthly_end_timestamp`

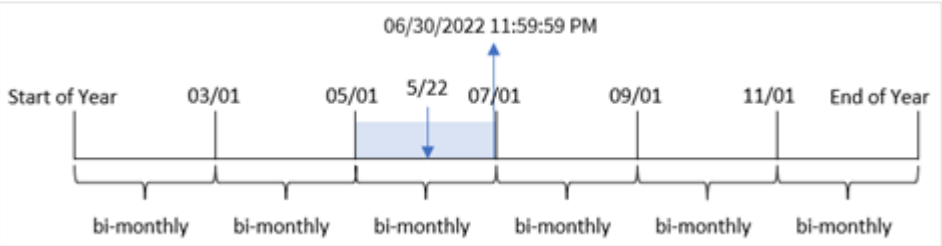
Tabela de resultados

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

id	date	bi_monthly_end	bi_monthly_end_timestamp
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

O campo "bi_monthly_end" é criado na instrução de carregamento anterior usando a função monthsend(). O primeiro argumento fornecido é 2, dividindo o ano em segmentos bimestrais. O segundo argumento identifica qual campo está sendo avaliado.

Diagrama da função monthsend com segmentos bimestrais.



A transação 8195 ocorre em 22 de maio. A função monthsend() inicialmente divide o ano em segmentos bimestrais. A transação 8195 cai no segmento entre 1º de maio e 30 de junho. Como resultado, a função retorna o último milissegundo desse segmento, 06/30/2022 11:59:59 PM.

Exemplo 2: period_no

Script de carregamento e resultados

Visão geral

São usados o mesmo conjunto de dados e cenário do primeiro exemplo.

Neste exemplo, a tarefa é criar um campo, “prev_bi_monthly_end”, que retorna o primeiro milissegundo do segmento bimestral antes de a transação ocorrer.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `id`
- `date`
- `prev_bi_monthly_end`
- `prev_bi_monthly_end_timestamp`

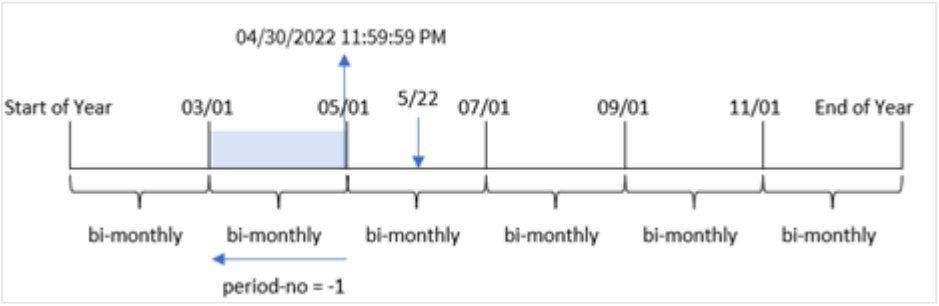
Tabela de resultados

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

id	date	prev_bi_monthly_end	prev_bi_monthly_end_timestamp
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

Usando `-1` como argumento `period_no` na função `monthsend()`, depois de dividir inicialmente um ano em segmentos bimestrais, a função retorna o último milissegundo do segmento bimestral anterior até quando uma transação ocorre.

Diagrama da função `monthsend` que retorna o segmento bimestral anterior.



A transação 8195 ocorre no segmento entre maio e junho. Como resultado, o segmento bimestral anterior estava entre 1º de março e 30 de abril e, portanto, a função retorna o último milissegundo desse segmento, 04/30/2022 11:59:59 PM.

Exemplo 3: first_month_of_year

Script de carregamento e resultados

Visão geral

São usados o mesmo conjunto de dados e cenário do primeiro exemplo.

Neste exemplo, a política organizacional é que abril seja o primeiro mês do exercício financeiro.

Crie um campo, "bi_monthly_end", que agrupa as transações em segmentos bimestrais e retorna o carimbo de data/hora do último milissegundo do segmento para cada transação.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- id
- date

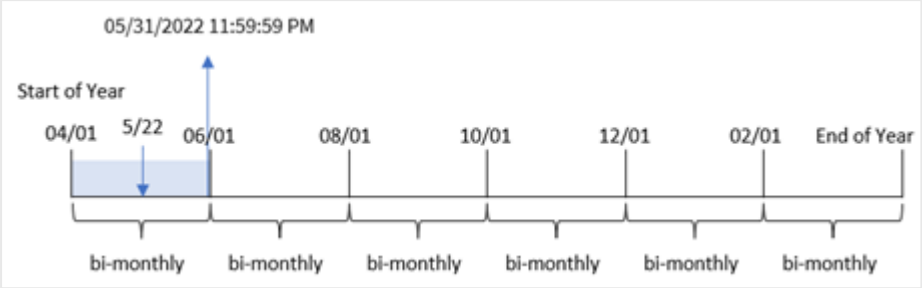
- bi_monthly_end
- bi_monthly_end_timestamp

Tabela de resultados

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
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

Usando 4 como argumento `first_month_of_year` na função `monthsend()`, a função começa o ano em 1º de abril. Em seguida, ela divide o ano em segmentos bimestrais: Apr-May, Jun-Jul, Aug-Sep, Oct-Nov, Dec-Jan, Feb-Mar.

Diagrama da função `monthsend` com o primeiro mês do ano definido como abril



A transação 8195 ocorreu em 22 de maio e cai no segmento entre 1º de abril e 31 de maio. Como resultado, a função retorna o último milissegundo desse segmento, 05/31/2022 11:59:59 PM.

Exemplo 4: Exemplo de objeto de gráfico

Script de carregamento e expressão de gráfico

Visão geral

São usados o mesmo conjunto de dados e cenário do primeiro exemplo. No entanto, neste exemplo, o conjunto de dados permanece inalterado e carregado no aplicativo.

Neste exemplo, a tarefa é criar um cálculo que agrupe as transações em segmentos bimestrais e retorne o carimbo de data/hora do último milissegundo do segmento para cada transação como uma medida em um objeto de gráfico de um aplicativo.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão:

`date`

Para obter o carimbo de data/hora do último milissegundo do segmento bimestral quando a transação ocorreu, crie as seguintes medidas:

- `=monthsEnd(2,date)`
- `=timestamp(monthsend(2,date))`

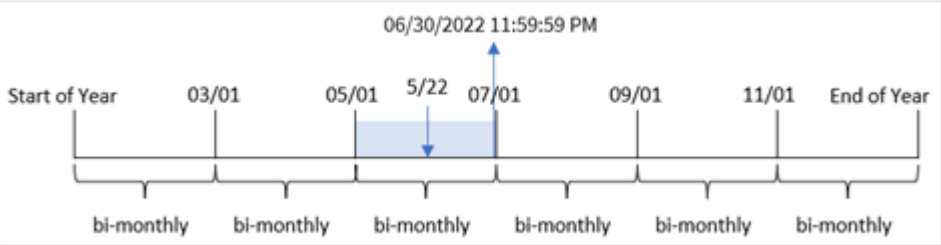
Tabela de resultados

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
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

id	date	=monthsend(2,date)	=timestamp(monthsend(2,date))
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

O campo “bi_monthly_end” é criado como uma medida no objeto de gráfico usando a função `monthsend()`. O primeiro argumento fornecido é `2`, que divide o ano em segmentos bimestrais. O segundo argumento identifica qual campo está sendo avaliado.

Diagrama da função `monthsend` com segmentos bimestrais.



A transação 8195 ocorre em 22 de maio. A função `monthsend()` inicialmente divide o ano em segmentos bimestrais. A transação 8195 cai no segmento entre 1º de maio e 30 de junho. Como resultado, a função retorna o primeiro milissegundo desse segmento, 06/30/2022 11:59:59 PM.

Exemplo 5: Cenário

Script de carregamento e resultados

Visão geral

Abra o editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia.

Neste exemplo, um conjunto de dados é carregado em uma tabela chamada "`Employee_Expenses`". A tabela contém os seguintes campos:

- IDs de funcionários
- Nomes de funcionários
- As reivindicações de despesas médias diárias de cada funcionário.

O usuário final deseja um gráfico que mostre, por ID do funcionário e nome do funcionário, a declaração de despesas estimada para o restante de um período de sua escolha. O exercício financeiro começa em janeiro.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma nova pasta.

No início do script de carregamento, é criada uma variável `vPeriod` que é vinculada ao controle de entrada da variável.

Faça o seguinte:

1. No painel de ativos, clique em **Objetos personalizados**.
2. Selecione o **pacote Qlik Dashboard**, crie um objeto de **Entrada de variável**.
3. Insira um título para o objeto de gráfico.
4. Em **Variável**, selecione **vPeriod** como o nome e defina o objeto para ser exibido como um **Menu suspenso**.
5. Em **Valores**, clique em valores **Dinâmicos**. Insira o seguinte: `='1~month|2~bi-month|3~quarter|4~tertial|6~half-year'.`

Crie uma nova tabela e esses campos como dimensões:

- `employee_id`
- `employee_name`

Para calcular os juros acumulados, crie esta medida:

```
=floor(monthsend($(vPeriod),today(1))-today(1))*avg_daily_claim
```

Essa medida é dinâmica e produzirá resultados de tabela diferentes, dependendo da data em que você carrega os dados.

Defina o **Formato numérico** da medida como **Dinheiro**.

Tabela de resultados

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

A função `monthsend()` usa a entrada do usuário como primeiro argumento e a data de hoje como segundo argumento. Isso retorna a data de término do período selecionado pelo usuário. Em seguida, a expressão retorna o número de dias que permanecem no período selecionado, subtraindo a data de hoje dessa data de término.

Esse valor é então multiplicado pela média de solicitações de despesas diárias por cada funcionário para calcular o valor estimado das solicitações que cada funcionário deve fazer nos dias restantes desse período.

monthsname

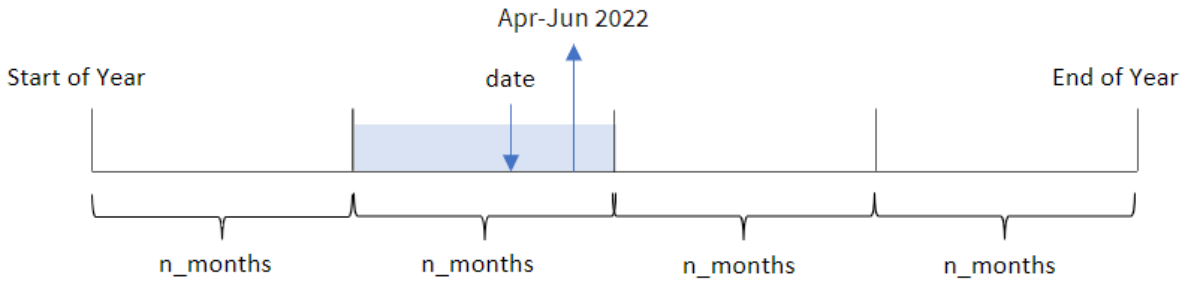
Esta função retorna um valor de exibição que representa o intervalo dos meses do período (formatados de acordo com a variável de script **MonthNames**) e o ano. O valor numérico subjacente corresponde a um carimbo de data/hora do primeiro milissegundo do mês, bimestre, trimestre, quadrimestre ou semestre contendo uma data base.

Sintaxe:

```
MonthsName(n_months, date[, period_no[, first_month_of_year]])
```

Tipo de dados de retorno: dual

Diagrama da função monthsname



A função `monthsname()` divide o ano em segmentos com base no argumento `n_months` fornecido. Em seguida, ela avalia o segmento ao qual cada `date` fornecida pertence e retorna os nomes dos meses inicial e final desse segmento, bem como o ano. A função também fornece a capacidade de retornar esses limites de segmentos anteriores ou seguintes, bem como de redefinir qual é o primeiro mês do ano.

Os seguintes segmentos do ano estão disponíveis na função como argumentos `n_month`:

Possíveis argumentos `n_month`

Períodos	Número de meses
mês	1
bimestre	2
trimestre	3
quadrimestre	4
semestre	6

Argumentos

Argumento	Descrição
n_months	O número de meses que define o período. Um inteiro ou uma expressão que se resolve como um inteiro que deve ser um dos seguintes: 1 (equivalente à função inmonth()), 2 (bimestre), 3 (equivalente à função inquarter()), 4 (quadrimestre) ou 6 (semestre).
date	A data ou o carimbo de data/hora a ser avaliado.
period_no	O período pode ser deslocado por **period_no**, um inteiro ou expressão que resolve um inteiro, no qual o valor 0 indica o período que contém **base_date**. Valores negativos em **period_no** indicam períodos precedentes e valores positivos indicam períodos sucessivos.
first_month_of_year	Se desejar trabalhar com anos (fiscais) que não comecem em janeiro, indique um valor entre 2 e 12 em **first_month_of_year**.

Quando usar

A função `monthsname()` é útil quando você deseja fornecer ao usuário a funcionalidade de comparar agregações por um período de sua escolha. Por exemplo, você pode fornecer uma variável de entrada para permitir que o usuário veja o total de vendas de produtos por mês, trimestre ou semestre.

Essas dimensões podem ser criadas no script de carregamento, adicionando a função como um campo em uma tabela Calendário mestre ou, como alternativa, criando a dimensão diretamente em um gráfico como uma dimensão calculada.

Exemplos de funções

Exemplo	Resultado
`monthsname(4, '10/19/2013')`	Retorna "Sep-Dec 2013". Neste e em outros exemplos, a instrução **SET Monthnames** está definida como Jan;Feb;Mar e assim por diante.
`monthsname(4, '10/19/2013', -1)`	Retorna "May-Aug 2013".
`monthsname(4, '10/19/2013', 0, 2)`	Retorna "Oct-Jan 2014", já que o ano está especificado para começar no mês 2. Portanto, o período de quatro meses termina no primeiro mês do ano seguinte.

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas

para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplo 1: Exemplo básico

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações para 2022, que é carregado em uma tabela denominada `Transactions`.
- O campo de data fornecido no formato da variável de sistema `DateFormat`(MM/DD/AAAA).
- A criação de um campo, `bi_monthly_range`, que agrupa transações em segmentos bimestrais e retorna os nomes dos limites desse segmento para cada transação.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

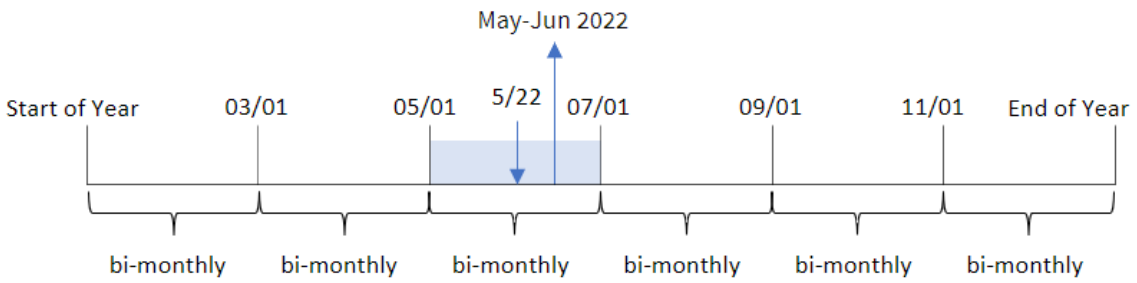
- `date`
- `bi_monthly_range`

Tabela de resultados

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

O campo `bi_monthly_range` é criado na instrução de carregamento anterior usando a função `monthsname()`. O primeiro argumento fornecido é 2, dividindo o ano em segmentos bimestrais. O segundo argumento identifica qual campo está sendo avaliado.

Diagrama da função monthsname, exemplo básico



A transação 8195 ocorre em 22 de maio. A função `monthsname()` inicialmente divide o ano em segmentos bimestrais. A transação 8195 cai no segmento entre 1º de maio e 30 de junho. Portanto, a função retorna esses meses no formato da variável do sistema `MonthNames`, bem como o ano, May-Jun 2022.

Exemplo 2: period_no

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- O mesmo conjunto de dados inline e cenário do primeiro exemplo.
- A criação de um campo, `prev_bi_monthly_range`, que agrupa transações em segmentos bimestrais e retorna os nomes dos limites do segmento anterior para cada transação.

Adicione seu outro texto aqui, conforme necessário, com listas etc.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `prev_bi_monthly_range`

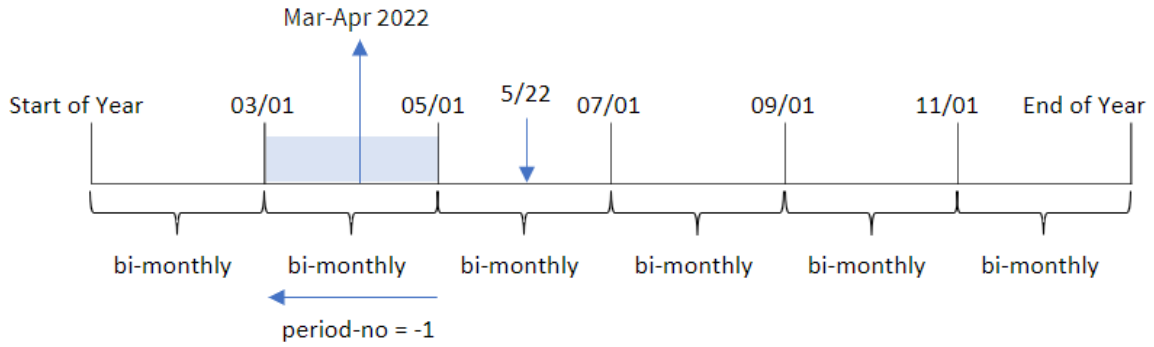
Tabela de resultados

date	**prev_bi_monthly_range**
2/19/2022	Nov-dez 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
8/2/2022	May-Jun 2022

date	prev_bi_monthly_range
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

Neste exemplo, -1 é usado como o argumento `period_no` na função `monthsname()`. Depois de dividir inicialmente um ano em segmentos bimestrais, a função retorna os limites do segmento anterior para quando uma transação ocorre.

Diagrama da função monthsname, exemplo de period_no



A transação 8195 ocorre no segmento entre maio e junho. Portanto, o segmento bimestral anterior foi entre 1º de março e 30 de abril, o que faz a função retornar Mar-abr 2022.

Exemplo 3: first_month_of_year

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- O mesmo conjunto de dados inline e cenário do primeiro exemplo.
- A criação de um campo diferente, `bi_monthly_range`, que agrupa as transações em segmentos bimestrais e retorna os limites do segmento para cada transação.

No entanto, neste exemplo, também precisamos definir abril como o primeiro mês do exercício financeiro.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `bi_monthly_range`

Tabela de resultados

date	bi_monthly_range
2/19/2022	Fev-mar 2021
3/7/2022	Fev-mar 2021
3/30/2022	Fev-mar 2021
4/5/2022	Apr-May 2022

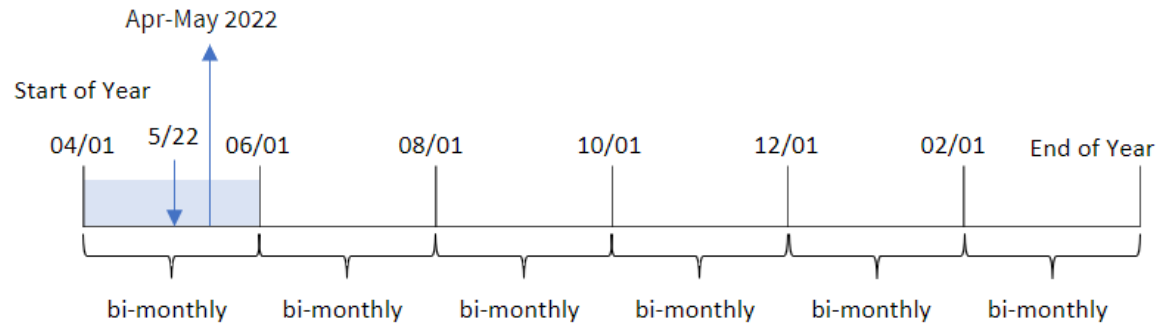
date	bi_monthly_range
4/16/2022	Apr-May 2022
5/1/2022	Apr-May 2022
5/7/2022	Apr-May 2022
5/22/2022	Apr-May 2022
6/15/2022	Jun-Jul 2022
6/26/2022	Jun-Jul 2022
7/9/2022	Jun-Jul 2022
7/22/2022	Jun-Jul 2022
7/23/2022	Jun-Jul 2022
7/27/2022	Jun-Jul 2022
8/2/2022	Aug-Sep 2022
8/8/2022	Aug-Sep 2022
8/19/2022	Aug-Sep 2022
9/26/2022	Aug-Sep 2022
10/14/2022	Oct-Nov 2022
10/29/2022	Oct-Nov 2022

Usando 4 como o argumento `first_month_of_year` na função `monthsname()`, a função começa o ano em 1º de abril. Em seguida, ela divide o ano em segmentos bimestrais: Abr-mai, Jun-jul, Ago-set, Out-nov, Dez-jan, Fev-mar.

Texto de parágrafo para resultados.

A transação 8195 ocorreu em 22 de maio e cai no segmento entre 1º de abril e 31 de maio. Portanto, a função retorna Apr-May 2022.

Diagrama da função monthsname, exemplo de first_month_of_year



Exemplo 4: Exemplo de objeto de gráfico

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém o mesmo conjunto de dados inline e cenário do primeiro exemplo. No entanto, neste exemplo, o conjunto de dados inalterado é carregado no aplicativo. O cálculo que agrupa transações em segmentos bimestrais e retorna os limites do segmento para cada transação é criado como uma medida em um objeto de gráfico do aplicativo.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão:`date`.

Crie a seguinte medida:

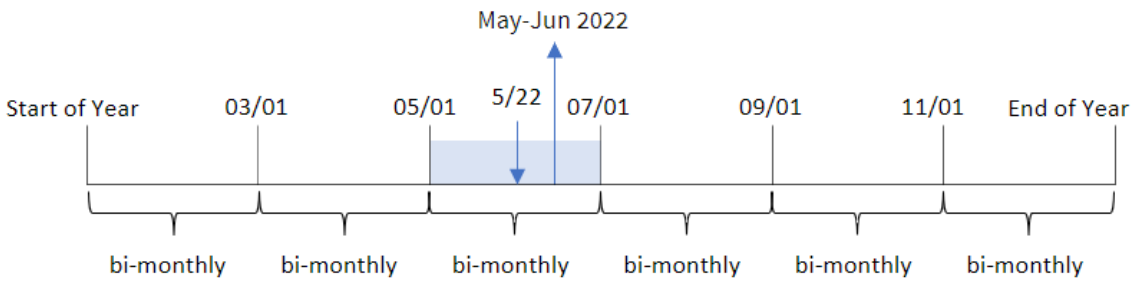
```
=monthsname(2,date)
```

Tabela de resultados

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

O campo `bi_monthly_range` é criado como uma medida no objeto de gráfico usando a função `monthsname()`. O primeiro argumento fornecido é 2, dividindo o ano em segmentos bimestrais. O segundo argumento identifica qual campo está sendo avaliado.

Diagrama da função monthsname, exemplo de objeto de gráfico



A transação 8195 ocorre em 22 de maio. A função `monthsname()` inicialmente divide o ano em segmentos bimestrais. A transação 8195 cai no segmento entre 1º de maio e 30 de junho. Portanto, a função retorna esses meses no formato da variável do sistema `MonthNames`, bem como o ano, May-Jun 2022.

Exemplo 5: Cenário

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo transações para 2022, que é carregado em uma tabela chamada `Transactions`.
- O campo de data fornecido no formato da variável de sistema `DateFormat` (MM/DD/AAAA).

O usuário final deseja ter um objeto de gráfico que exibe o total de vendas por um período de sua própria escolha. Isso pode ser alcançado mesmo quando essa dimensão não está disponível no modelo de dados, usando a função `monthsname()` como uma dimensão calculada que é modificada dinamicamente por um controle de entrada de variável.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta.

No início do script de carregamento, foi criada uma variável (`vPeriod`) que será vinculada ao controle de entrada de variável. Em seguida, configure a variável como um objeto personalizado na pasta.

Faça o seguinte:

1. No painel de ativos, clique em **Objetos personalizados**.
2. Selecione **Qlik - Pacote Dashboard** e crie um objeto de **Entrada de variável**.
3. Insira um título para o objeto de gráfico.
4. Em **Variável**, selecione **vPeriod** como o Nome e defina o objeto para ser exibido como um **Menu suspenso**.
5. Em **Valores**, configure o objeto para usar valores dinâmicos. Insira o seguinte:
   `='1~month|2~bi-month|3~quarter|4~tertial|6~half-year'`

Em seguida, crie a tabela de resultados.

Faça o seguinte:

1. Crie uma nova tabela e adicione a seguinte dimensão calculada:
   `=monthsname($(vPeriod),date)`
2. Adicione esta medida para calcular o total de vendas:
   `=sum(amount)`
3. Defina o **Formato numérico** da medida como **Dinheiro**. Clique em ✔ **Edição concluída**. Agora, você pode modificar os dados mostrados na tabela ajustando o segmento de tempo no objeto de variável.

É assim que a tabela de resultados ficará quando a opção `tertial` for selecionada:

Tabela de resultados

monthsname($(vPeriod),date)	=sum(amount)
Jan-Apr 2022	253.89
May-Aug 2022	713.58
Sep-Dec 2022	248.12

monthsstart

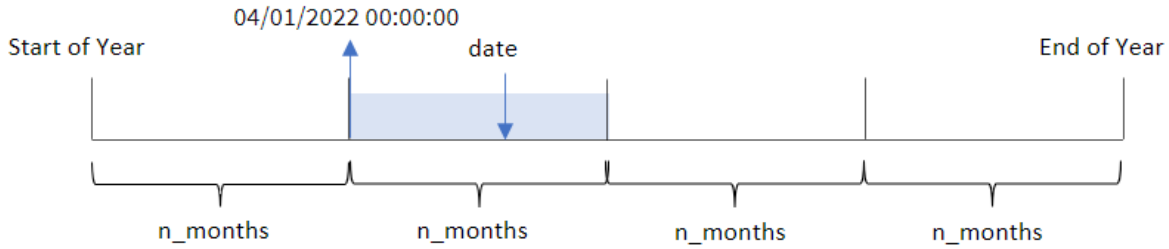
Essa função retorna um valor correspondente ao carimbo de data/hora do primeiro milissegundo do mês, bimestre, trimestre, quadrimestre ou semestre contendo uma data base. Também é possível descobrir a data/hora de um período de tempo anterior ou seguinte.O formato de saída padrão é o **DateFormat** definido no script.

Sintaxe:

```
MonthsStart(n_months, date[, period_no [, first_month_of_year]])
```

Tipo de dados de retorno: dual

Diagrama da função `monthsstart()`



A função `monthsstart()` divide o ano em segmentos com base no argumento `n_months` fornecido. Em seguida, ela avalia em qual segmento cada data fornecida se encaixa e retorna o primeiro milissegundo desse segmento, em formato de data. A função também fornece a capacidade de retornar o carimbo de data/hora inicial dos segmentos anteriores ou seguintes, bem como de redefinir qual é o primeiro mês do ano.

Os seguintes segmentos do ano estão disponíveis na função como argumentos `n_month`:

Possíveis argumentos `n_month`

Períodos	Número de meses
mês	1
bimestre	2
trimestre	3
quadrimestre	4
semestre	6

Argumentos

Argumento	Descrição
n_months	O número de meses que define o período. Um inteiro ou uma expressão que se resolve como um inteiro que deve ser um dos seguintes: 1 (equivalente à função inmonth()), 2 (bimestre), 3 (equivalente à função inquarter()), 4 (quadrimestre) ou 6 (semestre).
date	A data ou o carimbo de data/hora a ser avaliado.
period_no	O período pode ser deslocado por **period_no**, um inteiro ou expressão que resolve um inteiro, no qual o valor 0 indica o período que contém **base_date**. Valores negativos em **period_no** indicam períodos precedentes e valores positivos indicam períodos sucessivos.
first_month_of_year	Se desejar trabalhar com anos (fiscais) que não comecem em janeiro, indique um valor entre 2 e 12 em **first_month_of_year**.

Quando usar

A função `monthsstart()` é normalmente usada como parte de uma expressão quando o usuário deseja que o cálculo use a fração de um período que ainda não ocorreu. Isso pode ser usado, por exemplo, para fornecer uma variável de entrada para permitir que o usuário calcule o total de juros acumulados até o momento no mês, trimestre ou semestre.

Exemplos de funções

Exemplo	Resultado
`monthsstart(4, '10/19/2013')`	Retorna 09/01/2013.
`monthsstart(4, '10/19/2013, -1)`	Retorna 05/01/2013.
`monthsstart(4, '10/19/2013', 0, 2 )`	Retorna 10/01/2013, porque o início do ano se torna o mês 2.

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplo 1: Sem argumentos adicionais

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações para 2022, que é carregado em uma tabela denominada `Transactions`.
- O campo de data fornecido no formato da variável de sistema `DateFormat`(MM/DD/AAAA).
- A criação de um campo, `bi_monthly_start`, que agrupa as transações em segmentos bimestrais e retorna o carimbo de data/hora inicial do segmento para cada transação.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

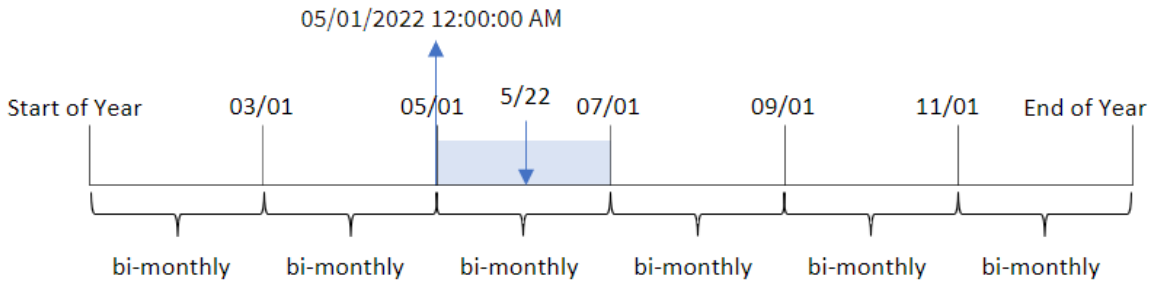
- `date`
- `bi_monthly_start`
- `bi_monthly_start_timestamp`

Tabela de resultados

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
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

O campo `bi_monthly_start` é criado na instrução de carregamento anterior usando a função `monthsstart()`. O primeiro argumento fornecido é 2, dividindo o ano em segmentos bimestrais. O segundo argumento identifica qual campo está sendo avaliado.

Diagrama da função `monthsstart()`, exemplo sem argumentos adicionais



A transação 8195 ocorre em 22 de maio. A função `monthsstart()` inicialmente divide o ano em segmentos bimestrais. A transação 8195 cai no segmento entre 1º de maio e 30 de junho. Portanto, a função retorna o primeiro milissegundo desse segmento, 1º de maio de 2022 às 12:00:00.

Exemplo 2: period_no

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- O mesmo conjunto de dados e cenário do primeiro exemplo.
- A criação de um campo, `prev_bi_monthly_start`, que retorna o primeiro milissegundo do segmento bimestral antes da transação.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `prev_bi_monthly_start`
- `prev_bi_monthly_start_timestamp`

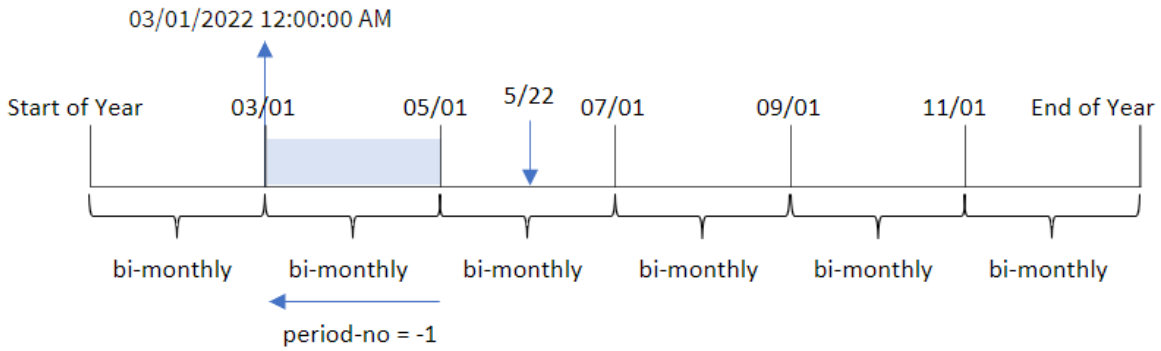
Tabela de resultados

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
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

date	prev_bi_monthly_start	prev_bi_monthly_start_timestamp
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

Usando -1 como o argumento `period_no` na função `monthsstart()`, depois de dividir inicialmente um ano em segmentos bimestrais, a função retorna o primeiro milissegundo do segmento bimestral anterior até quando uma transação ocorre.

Diagrama da função `monthsstart()`, exemplo de period_no



A transação 8195 ocorre no segmento entre maio e junho. Portanto, o segmento bimestral anterior estava entre 1º de março e 30 de abril, então a função retorna o primeiro milissegundo desse segmento, 1º de março de 2022, às 12:00:00 AM.

Exemplo 3: first_month_of_year

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- O mesmo conjunto de dados e cenário do primeiro exemplo.
- A criação de um campo, `bi_monthly_start`, que agrupa as transações em segmentos bimestrais e retorna o carimbo de data/hora inicial do conjunto para cada transação.

No entanto, neste exemplo, também precisamos definir abril como o primeiro mês do exercício financeiro.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `bi_monthly_start`
- `bi_monthly_start_timestamp`

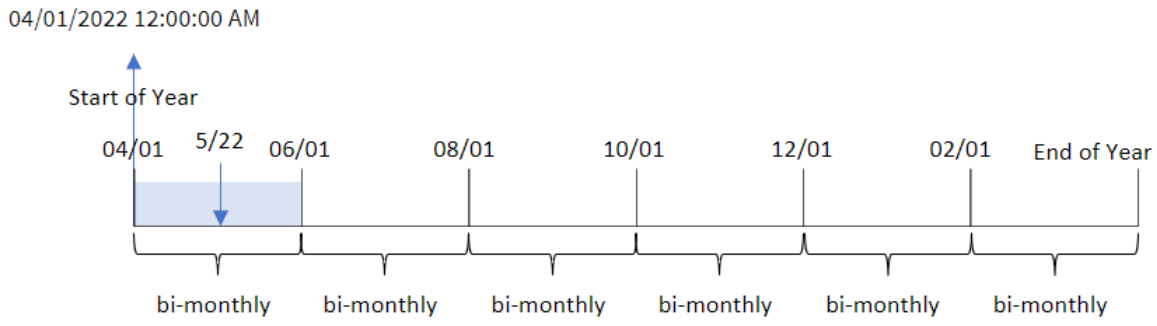
Tabela de resultados

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

date	bi_monthly_start	bi_monthly_start_timestamp
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

Usando 4 como o argumento `first_month_of_year` na função `monthsstart()`, a função começa o ano em 1º de abril. Em seguida, ela divide o ano em segmentos bimestrais: Abr-mai, Jun-jul, Ago-set, Out-nov, Dez-jan, Fev-mar.

Diagrama da função `monthsstart()`, exemplo de first_month_of_year



A transação 8195 ocorreu em 22 de maio e cai no segmento entre 1º de abril e 31 de maio. Portanto, a função retorna o primeiro milissegundo desse segmento, 1º de abril de 2022 às 12:00:00.

Exemplo 4: Exemplo de objeto de gráfico

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém o mesmo conjunto de dados e cenário do primeiro exemplo.

No entanto, neste exemplo, o conjunto de dados inalterado é carregado no aplicativo. O cálculo que agrupa as transações em segmentos bimestrais e retorna o carimbo de data/hora inicial do conjunto para cada transação é criado como uma medida em um objeto de gráfico do aplicativo.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão: `date`.

Crie as seguintes medidas:

```
=monthsstart(2,date)
```

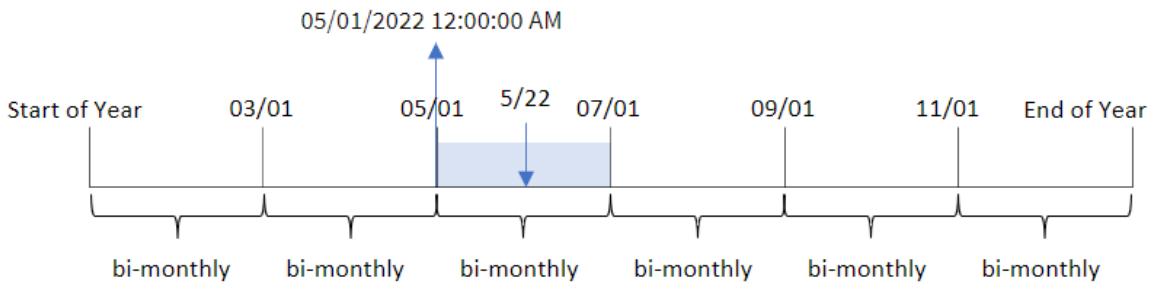
```
=timestamp(monthsstart(2,date))
```

Esses cálculos recuperarão o carimbo de data/hora inicial do segmento bimestral em que cada transação ocorreu.

Tabela de resultados

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
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

Diagrama da função `monthsstart()`, exemplo de objeto de gráfico



A transação 8195 ocorreu em 22 de maio. A função `monthsstart()` inicialmente divide o ano em segmentos bimestrais. A transação 8195 cai no segmento entre 1º de maio e 30 de junho. Portanto, a função retorna o primeiro milissegundo desse segmento, 05/01/2022 12:00:00 AM.

Exemplo 5: Cenário

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de saldos de empréstimos, que é carregado em uma tabela denominada `Loans`.
- Dados que consistem em IDs de empréstimos, saldo no início do mês e taxa de juros simples cobrada em cada empréstimo por ano.

O usuário final gostaria de um objeto de gráfico que mostre, por ID de empréstimo, os juros atuais acumulados em cada empréstimo no período de sua escolha. O exercício financeiro começa em janeiro.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta.

No início do script de carregamento, foi criada uma variável (`vPeriod`) que será vinculada ao controle de entrada de variável. Em seguida, configure a variável como um objeto personalizado na pasta.

Faça o seguinte:

1. No painel de ativos, clique em **Objetos personalizados**.
2. Selecione **Qlik - Pacote Dashboard** e crie um objeto de **Entrada de variável**.
3. Insira um título para o objeto de gráfico.
4. Em **Variável**, selecione **vPeriod** como o Nome e defina o objeto para ser exibido como um **Menu suspenso**.
5. Em **Valores**, configure o objeto para usar valores dinâmicos. Insira o seguinte:
   `='1~month|2~bi-month|3~quarter|4~tertial|6~half-year'`

Em seguida, crie a tabela de resultados.

Faça o seguinte:

1. Crie uma nova tabela. Adicione os seguintes campos como dimensões.
   - `employee_id`
   - `employee_name`
2. Crie uma medida para calcular os juros acumulados:
   `=start_balance*(rate*(today(1)-monthsstart($(vPeriod),today(1)))/365)`
3. Defina o **Formato numérico** da medida como **Dinheiro**. Clique em ✔ **Edição concluída**. Agora, você pode modificar os dados mostrados na tabela ajustando o segmento de tempo no objeto de variável.

É assim que a tabela de resultados ficará quando a opção de período `month` for selecionada:

Tabela de resultados

loan_id	start_balance	=start_balance*(rate*(today(1)-monthsstart($(vPeriod),today(1)))/365)
8188	$10000.00	$7.95
8189	$15000.00	$67.93
8190	$17500.00	$33.37
8191	$21000.00	$56.73
8192	$90000.00	$600.66

A função `monthsstart()`, usando a entrada do usuário como primeiro argumento e a data de hoje como segundo argumento, retorna a data de início do período escolhido pelo usuário. Ao subtrair esse resultado da data atual, a expressão retorna o número de dias decorridos até o momento nesse período.

Esse valor é então multiplicado pela taxa de juros e dividido por 365 para retornar a taxa de juros efetiva acumulada do período. O resultado é então multiplicado pelo saldo inicial do empréstimo para retornar os juros acumulados até o momento nesse período.

monthstart

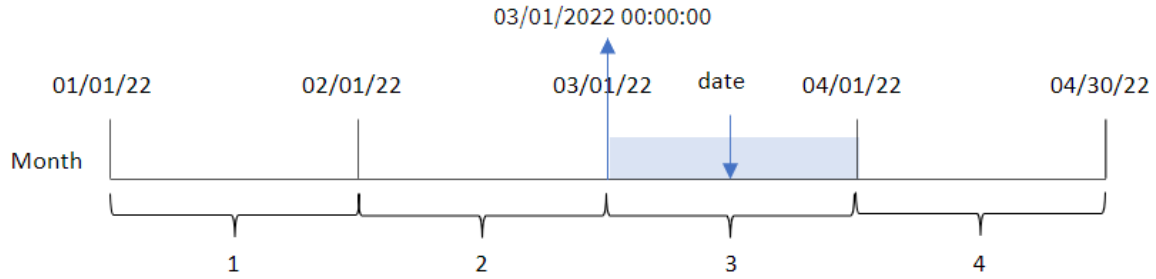
Esta função retorna um valor correspondente à data/hora com o primeiro milissegundo do primeiro dia do mês que contém **date**. O formato de saída padrão será o **DateFormat** definido no script.

Sintaxe:

```
MonthStart(date[, period_no])
```

Tipo de dados de retorno: dual

Diagrama da função monthstart()



A função `monthstart()` determina em qual mês a data cai. Em seguida, ela retorna um carimbo de data/hora, em formato de data, para o primeiro milissegundo daquele mês.

Argumentos

Argumento	Descrição
date	A data ou o carimbo de data/hora a ser avaliado.
period_no	**period_no** é um inteiro, no qual, se 0 ou omitido, indica o mês que contém **date**. Valores negativos em **period_no** indicam meses precedentes e valores positivos indicam meses sucessivos.

Quando usar

A função `monthstart()` é comumente usada como parte de uma expressão quando o usuário deseja que o cálculo use a fração do mês decorrido até o momento. Por exemplo, ela pode ser usada para calcular os juros acumulados em um mês até uma determinada data.

Exemplos de funções

Exemplo	Resultado
`monthstart('10/19/2001')`	Retorna 10/01/2001.
`monthstart('10/19/2001', -1)`	Retorna 09/01/2001.

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplo 1: Sem argumentos adicionais

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações para 2022, que é carregado em uma tabela denominada `Transactions`.
- O campo de data fornecido no formato da `variável` de sistema DateFormat (MM/DD/AAAA).
- A criação de um campo, `start_of_month`, que retorna um carimbo de data/hora para o início do mês em que as transações ocorreram.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `start_of_month`
- `start_of_month_timestamp`

Tabela de resultados

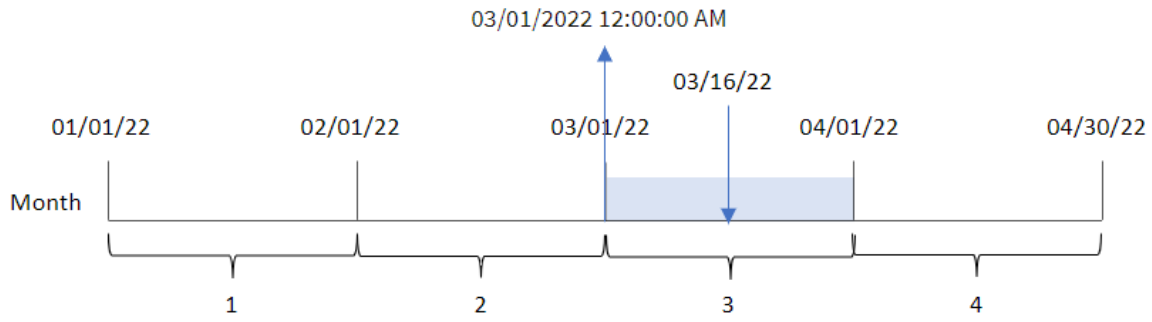
date	start_of_month	start_of_month_timestamp
1/7/2022	01/01/2022	1/1/2022 12:00:00 AM
1/19/2022	01/01/2022	1/1/2022 12:00:00 AM
2/5/2022	02/01/2022	2/1/2022 12:00:00 AM
2/28/2022	02/01/2022	2/1/2022 12:00:00 AM
3/16/2022	03/01/2022	3/1/2022 12:00:00 AM
4/1/2022	04/01/2022	4/1/2022 12:00:00 AM
5/7/2022	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

date	start_of_month	start_of_month_timestamp
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

O campo `start_of_month` é criado na instrução de carregamento anterior usando a função `monthstart()` e transmitindo o campo de data como o argumento da função.

A função `monthstart()` identifica em qual mês o valor da data cai, retornando um carimbo de data/hora para o primeiro milissegundo desse mês.

Diagrama da função `monthstart()`, exemplo sem argumentos adicionais



A transação 8192 ocorreu em 16 de março. A função `monthstart()` retorna o primeiro milissegundo desse mês, que é 1º de março às 12:00:00 AM.

Exemplo 2: period_no

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- O mesmo conjunto de dados e cenário do primeiro exemplo.
- A criação de um campo, `previous_month_start`, que retorna o carimbo de data/hora inicial do mês antes da transação.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- date
- previous_month_start
- previous_month_start_timestamp

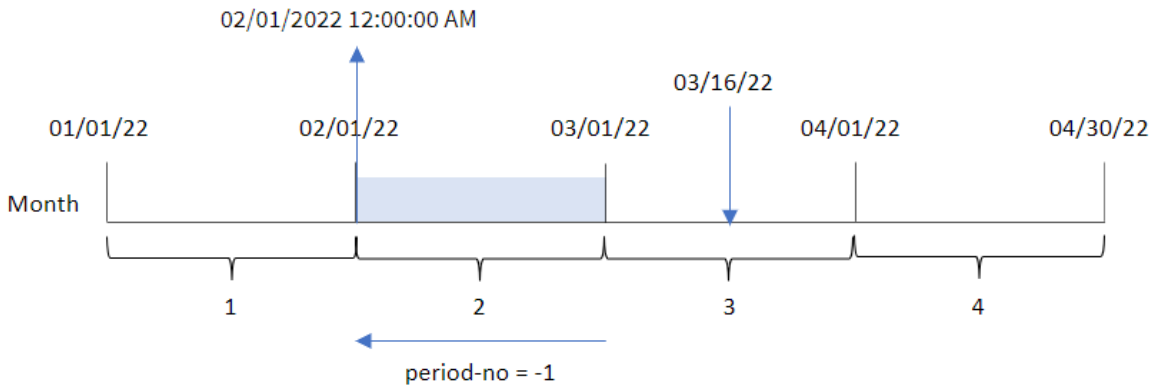
Tabela de resultados

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

date	previous_month_start	previous_month_start_timestamp
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
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

Nesse caso, como um `period_no` de `-1` foi usado como o argumento offset na função `monthstart()`, a função primeiro identifica o mês em que as transações ocorrem. Em seguida, ela muda um mês antes e identifica o primeiro milissegundo desse mês.

Diagrama da função `monthstart()`, exemplo de period_no



A transação 8192 ocorreu em 16 de março. A função `monthstart()` identifica que o mês anterior à realização da transação foi em fevereiro. Em seguida, ela retorna o primeiro milissegundo desse mês, 1º de fevereiro, às 12h00.

Exemplo 3: Exemplo de objeto de gráfico

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém o mesmo conjunto de dados e cenário do primeiro exemplo.

No entanto, neste exemplo, o conjunto de dados inalterado é carregado no aplicativo. O cálculo que retorna um carimbo de data/hora para o início do mês em que as transações ocorreram é criado como uma medida em um objeto de gráfico do aplicativo.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão: `date`.

Para calcular a data de início do mês em que uma transação ocorre, crie as seguintes medidas:

- `=monthstart(date)`
- `=timestamp(monthstart(date))`

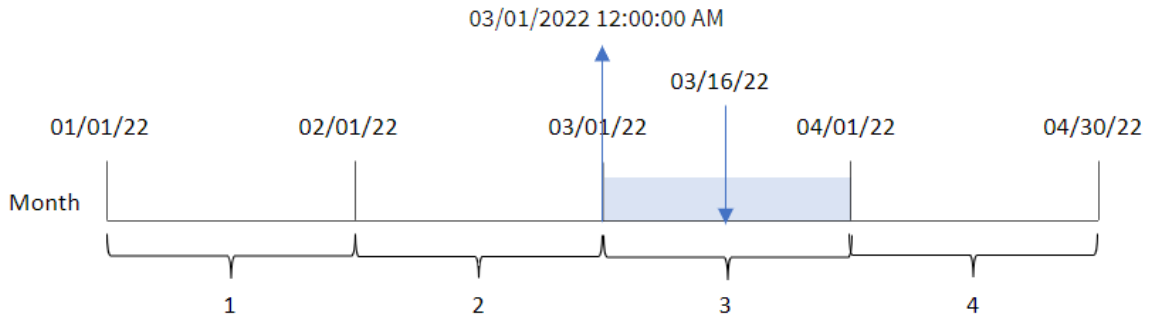
Tabela de resultados

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
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

A medida `start_of_month` é criada no objeto de gráfico usando a função `monthstart()` e transmitindo o campo de data como o argumento da função.

A função `monthstart()` identifica em qual mês o valor da data cai, retornando um carimbo de data/hora para o primeiro milissegundo desse mês.

Diagrama da função `monthstart()`, exemplo de objeto de gráfico



A transação 8192 ocorreu em 16 de março. A função `monthstart()` identifica que a transação ocorreu em março e retorna o primeiro milissegundo desse mês, que é 1º de março às 12:00:00 AM.

Exemplo 4: Cenário

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de saldos de empréstimos, que é carregado em uma tabela denominada `Loans`.
- Dados que consistem em IDs de empréstimos, saldo no início do mês e taxa de juros simples cobrada em cada empréstimo por ano.

O usuário final deseja um objeto de gráfico que mostre, por ID de empréstimo, os juros atuais que foram acumulados em cada empréstimo no mês até o momento.

Script de carregamento

```
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
];
```

Resultados

Faça o seguinte:

1. Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:
   - `loan_id`
   - `start_balance`
2. Em seguida, crie uma medida para calcular os juros acumulados:
   `=start_balance*(rate*(today(1)-monthstart(today(1)))/365)`
3. Defina o **Formato numérico** da medida como **Dinheiro**.

Tabela de resultados

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

A função `monthstart()`, usando a data de hoje como seu único argumento, retorna a data de início do mês atual. Ao subtrair esse resultado da data atual, a expressão retorna o número de dias decorridos até o momento neste mês.

Esse valor é então multiplicado pela taxa de juros e dividido por 365 para retornar a taxa de juros efetiva acumulada do período. O resultado é então multiplicado pelo saldo inicial do empréstimo para retornar os juros acumulados até o momento neste mês.

networkdays

A função **networkdays** retorna o número de dias úteis (segunda-sexta) entre e inclusive a **start_date** e **end_date**, levando em conta qualquer **holiday** opcionalmente listado.

Sintaxe:

```
networkdays (start_date, end_date [, holiday])
```

Tipo de dados de retorno: inteiro

Diagrama de calendário exibindo o intervalo de datas retornado pela função networkdays

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			

A função `networkdays` tem as seguintes limitações:

- Não há método para modificar os dias úteis. Em outras palavras, não há como modificar a função para regiões ou situações que envolvam algo além de trabalhar de segunda a sexta-feira.
- O parâmetro `holiday` deve ser uma constante de cadeia de caracteres. Expressões não são aceitas.

Argumentos

Argumento	Descrição
start_date	A data inicial para avaliar.
end_date	A data final para avaliar.
holiday	Períodos de feriados a serem excluídos dos dias de trabalho. Um feriado é declarado como uma data constante de string. Você pode especificar várias datas de feriados, separadas por vírgulas. **Exemplo:** `'12/25/2013', '12/26/2013', '12/31/2013', '01/01/2014'`

Quando usar

A função `networkdays()` é normalmente usada como parte de uma expressão quando o usuário deseja que o cálculo use o número de dias úteis da semana que ocorrem entre duas datas. Por exemplo, se um usuário quiser calcular o total de salários que serão ganhos por um funcionário em um contrato PAYE (pague conforme você ganha).

Exemplos de funções

Exemplo	Resultado
`networkdays ('12/19/2013', '01/07/2014')`	Retorna 14. Este exemplo não considera feriados.
`networkdays ('12/19/2013', '01/07/2014', '12/25/2013', '12/26/2013')`	Retorna 12. Este exemplo não considera o feriado de 12/25/2013 a12/26/2013.
`networkdays ('12/19/2013', '01/07/2014', '12/25/2013', '12/26/2013', '12/31/2013', '01/01/2014')`	Retorna 10. Este exemplo considera dois períodos de feriados.

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplo 1: Exemplo básico

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo IDs de projetos, suas datas de início e suas datas de término. Essas informações são carregadas em uma tabela denominada `Projects`.
- O campo de data fornecido no formato da `variável` de sistema DateFormat (MM/DD/AAAA).
- A criação de um campo adicional, `net_work_days`, para calcular o número de dias úteis envolvidos em cada projeto.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- id
- start_date
- end_date
- net_work_days

Tabela de resultados

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

Como não há feriados programados (isso estaria presente no terceiro argumento da função networkdays()), a função também subtrai o start_date de end_date como todos os fins de semana, para calcular o número de dias úteis entre as duas datas.

Diagrama de calendário destacando dias úteis para o projeto 5 (sem feriados)

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			

O calendário acima descreve visualmente o projeto com `id` de 5. O projeto 5 começa na quarta-feira, 10 de agosto de 2022, e termina em 26 de agosto de 2022. Com todos os sábados e domingos ignorados, há 13 dias úteis entre, e inclusive, essas duas datas.

Exemplo 2: Feriado único

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- O mesmo conjunto de dados e cenário do exemplo anterior.
- O campo de data fornecido no formato da variável de sistema `DateFormat` (MM/DD/AAAA).
- A criação de um campo adicional, `net_work_days`, para calcular o número de dias úteis envolvidos em cada projeto.

Neste exemplo, há um feriado de um dia agendado para 19 de agosto de 2022.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- id
- start_date
- end_date
- net_work_days

Tabela de resultados

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

O único feriado agendado é inserido como o terceiro argumento na função `networkdays()`.

Diagrama de calendário destacando os dias úteis para o projeto 5 (feriado único)

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			

O calendário acima descreve visualmente o projeto 5, demonstrando esse ajuste para incluir o feriado. Esse feriado ocorre durante o projeto 5 na sexta-feira, 19 de agosto de 2022. Como resultado, o valor `net_work_days` total do projeto 5 diminui em um dia, de 13 para 12 dias.

Exemplo 3: Vários feriados

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- O mesmo conjunto de dados e cenário do primeiro exemplo.
- O campo de data fornecido no formato da variável de sistema `DateFormat` (MM/DD/AAAA).
- A criação de um campo adicional, `net_work_days`, para calcular o número de dias úteis envolvidos em cada projeto.

No entanto, neste exemplo, há quatro feriados agendados de 18 a 21 de agosto de 2022.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- id
- start_date
- end_date
- net_work_days

Tabela de resultados

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

Os quatro feriados agendados são inseridos como uma lista separada por vírgulas, a partir do terceiro argumento na função networkdays().

Diagrama de calendário destacando os dias úteis para o projeto 5 (vários feriados)

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			

O calendário acima descreve visualmente o projeto 5, demonstrando esse ajuste para incluir esses feriados. Esse período de feriados agendados ocorre durante o projeto 5, com dois dos dias ocorrendo em uma quinta e sexta-feira. Como resultado, o valor `net_work_days` total do projeto 5 diminui de 13 para 11 dias.

Exemplo 4: Feriado único

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- O mesmo conjunto de dados e cenário do primeiro exemplo.
- O campo de data fornecido no formato da variável de sistema `DateFormat` (MM/DD/AAAA).

Há um feriado de um dia agendado para 19 de agosto de 2022.

No entanto, neste exemplo, o conjunto de dados inalterado é carregado no aplicativo. O campo `net_work_days` é calculado como uma medida em um objeto de gráfico.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `id`
- `start_date`
- `end_date`

Crie a seguinte medida:

```
= networkdays(start_date,end_date,'08/19/2022')
```

Tabela de resultados

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

O único feriado agendado é inserido como o terceiro argumento na função `networkdays()`.

Diagrama de calendário mostrando dias úteis líquidos com um único feriado (objeto de gráfico)

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			

O calendário acima descreve visualmente o projeto 5, demonstrando esse ajuste para incluir o feriado. Esse feriado ocorre durante o projeto 5 na sexta-feira, 19 de agosto de 2022. Como resultado, o valor `net_work_days` total do projeto 5 diminui em um dia, de 13 para 12 dias.

now

Essa função retorna um carimbo de data/hora da hora atual. A função retorna valores no formato da variável do sistema **TimeStamp**. O valor padrão de **timer_mode** é 1.

Sintaxe:

```
now([ timer_mode])
```

Tipo de dados de retorno: dual

A função `now()` pode ser usada no script de carregamento ou em objetos de gráfico.

Argumentos

Argumento	Descrição
timer_mode	Pode ter os seguintes valores: 0 (hora no último carregamento de dados finalizado) 1 (hora na chamada de função) 2 (hora em que o aplicativo foi aberto) *Se você usar a função em um script de carregamento de dados,* ***timer_mode=0*** *resultará na hora da última carga de dados acabada, enquanto que* ***timer_mode=1*** *dará a hora da chamada de função do carregamento de dados atual.*

A função `now()` tem um impacto de alto desempenho, o que pode resultar em problemas de rolagem se a função for usada nas expressões das tabelas. Sempre que seu uso não for estritamente necessário, recomendamos o uso da função `today()`. Caso seja necessário o uso de `now()` em um layout, recomendamos utilizar as configurações não padrão `now(0)` ou `now(2)` quando possível, pois não requerem recálculos constantes

Quando usar

A função `now()` é normalmente usada como um componente dentro de uma expressão. Por exemplo, ela pode ser usada para calcular o tempo restante no ciclo de vida de um produto. A função `now()` seria usada em vez da função `today()` quando a expressão requer o uso de uma fração de um dia.

A tabela a seguir fornece uma explicação do resultado retornado pela função `now()`, considerando valores diferentes para o argumento `timer_mode`:

Exemplos de funções

timer_mode value	Resultado se usado no script de carregamento	Resultado se usado no objeto de gráfico
0	Retorna um carimbo de data/hora, no formato da `variável` de sistema TimeStamp, da última recarga de dados bem-sucedida antes do último carregamento de dados.	Retorna um carimbo de data/hora, no formato da variável de sistema `TimeStamp`, para o último carregamento de dados.
1	Retorna um carimbo de data/hora, no formato da variável de sistema `TimeStamp`, para o último carregamento de dados.	Retorna um carimbo de data/hora, no formato da variável de sistema `TimeStamp`, da chamada da função.

timer_mode value	Resultado se usado no script de carregamento	Resultado se usado no objeto de gráfico
2	Retorna um carimbo de data/hora, no formato de variável de sistema `TimeStamp`, de quando a sessão do usuário no aplicativo começou. Isso não será atualizado a menos que o usuário recarregue o script.	Retorna o carimbo de data/hora, no formato de variável de sistema `TimeStamp`, de quando a sessão do usuário no aplicativo começou. Isso será atualizado quando uma nova sessão for iniciada ou quando os dados do aplicativo forem recarregados.

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplo 1: geração de objetos usando script de carregamento

Script de carregamento e resultados

Visão geral

Abra o editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia.

Este exemplo cria três variáveis usando a função `now()`. Cada variável usa uma das opções `timer_mode` para demonstrar seu efeito.

Para que as variáveis demonstrem sua finalidade, carregue o script e, depois de um curto período, carregue o script pela segunda vez. Isso fará com que as variáveis `now(0)` e `now(1)` mostrem valores diferentes, demonstrando corretamente sua finalidade.

Script de carregamento

```
LET vPreviousDataLoad = now(0);
LET vCurrentDataLoad = now(1);
LET vApplicationOpened = now(2);
```

Resultados

Depois que os dados tiverem sido carregados pela segunda vez, crie três caixas de texto usando as instruções abaixo.

Primeiro, crie uma caixa de texto para os dados que foram carregados anteriormente.

Faça o seguinte:

1. Usando o objeto de gráfico **Texto e imagem**, crie uma caixa de texto.
2. Adicione a seguinte medida ao objeto:
   `=vPreviousDataLoad`
3. Em **Aparência**, selecione **Show titles** e adicione o título "Tempo de carregamento anterior" ao objeto.

Em seguida, crie uma caixa de texto para os dados que estão sendo carregados.

Faça o seguinte:

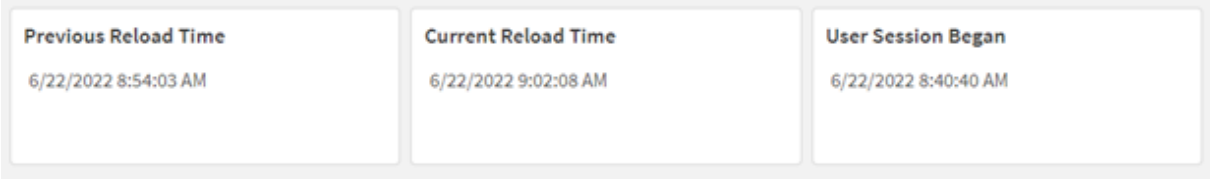
1. Usando o objeto de gráfico **Texto e imagem**, crie uma caixa de texto.
2. Adicione a seguinte medida ao objeto:
   `=vCurrentDataLoad`
3. Em **Aparência**, selecione **Show titles** e adicione o título "Tempo de carregamento atual" ao objeto.

Crie uma caixa de texto final para mostrar quando a sessão do usuário no aplicativo foi iniciada.

Faça o seguinte:

1. Usando o objeto de gráfico **Texto e imagem**, crie uma caixa de texto.
2. Adicione a seguinte medida ao objeto:
   `=vApplicationOpened`
3. Em **Aparência**, selecione **Show titles** e adicione o título "Sessão do usuário iniciada" ao objeto.

Variáveis de script de carregamento now()



A imagem acima mostra exemplos de valores para cada uma das variáveis criadas. Por exemplo, os valores podem ser os seguintes:

- Tempo de carregamento anterior: 6/22/2022 8:54:03 AM
- Tempo de carregamento atual: 6/22/2022 9:02:08 AM
- Início da sessão do usuário: 6/22/2022 8:40:40 AM

Exemplo 2: geração de objetos sem script de carregamento

Script de carregamento e expressão de gráfico

Visão geral

Neste exemplo, você criará três objetos gráficos usando a função `now()`, sem carregar nenhuma variável ou dado no aplicativo. Cada objeto de gráfico usa uma das opções `timer_mode` para demonstrar seu efeito.

Não há script de carregamento para esse exemplo.

Faça o seguinte:

1. Abra o editor de carregamento de dados.
2. Sem alterar o script de carregamento existente, clique em **Carregar dados**.
3. Depois de um curto período, carregue o script pela segunda vez.

Resultados

Depois que os dados tiverem sido carregados pela segunda vez, crie três caixas de texto.

Primeiro, crie uma caixa de texto para o último carregamento de dados.

Faça o seguinte:

1. Usando o objeto de gráfico **Texto e imagem**, crie uma caixa de texto.
2. Adicione a seguinte medida.
   `=now(0)`
3. Em **Aparência**, selecione **Mostrar títulos** e adicione o título "Carregamento de dados mais recente" ao objeto.

Em seguida, crie uma caixa de texto para mostrar a hora atual.

Faça o seguinte:

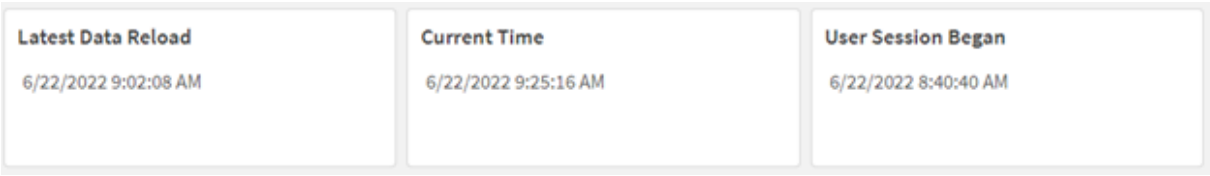
1. Usando o objeto de gráfico **Texto e imagem**, crie uma caixa de texto.
2. Adicione a seguinte medida:
   `=now(1)`
3. Em **Aparência**, selecione **Mostrar títulos** e adicione o título "Hora atual" ao objeto.

Crie uma caixa de texto final para mostrar quando a sessão do usuário no aplicativo foi iniciada.

Faça o seguinte:

1. Usando o objeto de gráfico **Texto e imagem**, crie uma caixa de texto.
2. Adicione a seguinte medida:
   `=now(2)`
3. Em **Aparência**, selecione **Mostrar títulos** e adicione o título “Início da sessão do usuário” ao objeto.

Exemplos de objetos de gráfico de now()



A imagem acima mostra exemplos de valores para cada um dos objetos criados. Por exemplo, os valores podem ser os seguintes:

- Último carregamento de dados: 6/22/2022 9:02:08 AM
- Hora atual: 6/22/2022 9:25:16 AM
- Início da sessão do usuário: 6/22/2022 8:40:40 AM

O objeto de gráfico “Carregamento de dados mais recente” usa um valor `timer_mode` de 0. Isso retorna o carimbo de data/hora da última vez em que os dados foram carregados com êxito.

O objeto de gráfico “Hora atual” usa um valor `timer_mode` de 1. Isso retorna a hora atual de acordo com o relógio do sistema. Se a pasta ou o objeto forem atualizados, esse valor será atualizado.

O objeto de gráfico “Início da sessão do usuário” usa um valor `timer_mode` de 2. Isso retorna o carimbo de data/hora de quando o aplicativo foi aberto e a sessão do usuário foi iniciada.

Exemplo 3: Cenário

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados que consiste no inventário de uma operação de mineração de criptomoedas, que é carregado em uma tabela denominada `Inventory`.
- Dados com os seguintes campos: `id`, `purchase_date` e `wph` (watts por hora).

O usuário deseja uma tabela que mostre, por `id`, o custo total que cada plataforma de mineração acumulou no mês até o momento, em termos de consumo de energia.

Esse valor deve ser atualizado sempre que o objeto de gráfico for atualizado. O custo atual da eletricidade é de $0,0678 por kWh.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão: `id`.

Crie a seguinte medida:

```
=(now(1)-monthstart(now(1)))*24*wph/1000*0.0678
```

Se o objeto de gráfico fosse atualizado em 6/22/2022 10:39:05 AM, ele retornaria os seguintes resultados:

Tabela de resultados

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

id	=(now(1)-monthstart(now(1)))*24*wph/1000*0.0678
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

O usuário gostaria que os resultados do objeto fossem atualizados sempre que esse objeto fosse atualizado. Portanto, o argumento `timer_mode` fornecido para instâncias da função `now()` na expressão. O carimbo de data/hora para o início do mês, identificado usando a função `now()` como o argumento timestamp na função `monthstart()`, é subtraído da hora atual que é identificada pela função `now()`. Isso fornece a quantidade total de tempo decorrido até o momento neste mês, em dias.

Esse valor é multiplicado por 24 (o número de horas em um dia) e depois pelo valor no `campo` wph.

Para converter de watts por hora em quilowatts por hora, o resultado é dividido por 1000 antes de finalmente ser multiplicado pela taxa de kWh fornecida.

quarterend

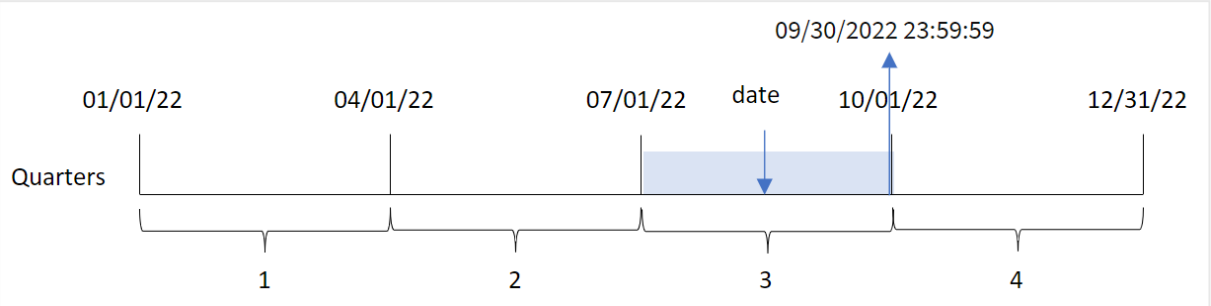
Esta função retorna um valor correspondente a uma data/hora com o último milissegundo do trimestre que contém **date**. O formato de saída padrão será o **DateFormat** definido no script.

Sintaxe:

```
QuarterEnd(date[, period_no[, first_month_of_year]])
```

Tipo de dados de retorno: dual

Diagrama da função quarterend()



A função quarterend() determina em qual trimestre a data cai. Em seguida, ele retorna um carimbo de data/hora, no formato de data, para o último milissegundo do último mês desse trimestre. O primeiro mês do ano é, por padrão, janeiro. No entanto, você pode alterar o mês definido como o primeiro usando o argumento first_month_of_year na função quarterend().

A função quarterend() não considera a variável de sistema FirstMonthOfYear. O ano começa em 1º de janeiro, a menos que o argumento first_month_of_year seja usado para alterá-lo.

Quando usar

A função quarterend() é normalmente usada como parte de uma expressão quando você deseja que o cálculo use a fração do trimestre que ainda não ocorreu. Por exemplo, se você quiser calcular o total de juros ainda não acumulados durante o trimestre.

Argumentos

Argumento	Descrição
date	A data ou o carimbo de data/hora a ser avaliado.
period_no	**period_no** é um inteiro, em que o valor 0 indica o trimestre que contém **date**. Valores negativos em **period_no** indicam trimestres precedentes e valores positivos indicam trimestres sucessivos.
first_month_of_year	Se desejar trabalhar com anos (fiscais) que não comecem em janeiro, indique um valor entre 2 e 12 em **first_month_of_year**.

Você pode usar os seguintes valores para definir o primeiro mês do ano no argumento first_month_of_year:

Valores first_month_of_year

Month	Valor
Fevereiro	2

Month	Valor
Março	3
Abril	4
Maio	5
Junho	6
Julho	7
Agosto	8
Setembro	9
Outubro	10
Novembro	11
Dezembro	12

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplos de funções

Exemplo	Resultado
`quarterend('10/29/2005')`	Retorna 12/31/2005 23:59:59.
`quarterend('10/29/2005', -1)`	Retorna 09/30/2005 23:59:59.
`quarterend('10/29/2005', 0, 3)`	Retorna 11/30/2005 23:59:59.

Exemplo 1: Exemplo básico

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações em 2022 que é carregado em uma tabela denominada "Transactions".
- Um carregamento anterior que contém o seguinte:
    - A função quarterend() que é definida como o campo “end_of_quarter” e retorna um carimbo de data/hora para o final do trimestre em que as transações ocorreram.
    - A função timestamp() que é definida como o campo “end_of_quarter_timestamp” e retorna o carimbo de data/hora exato do final do trimestre selecionado.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- id
- date
- end_of_quarter
- end_of_quarter_timestamp

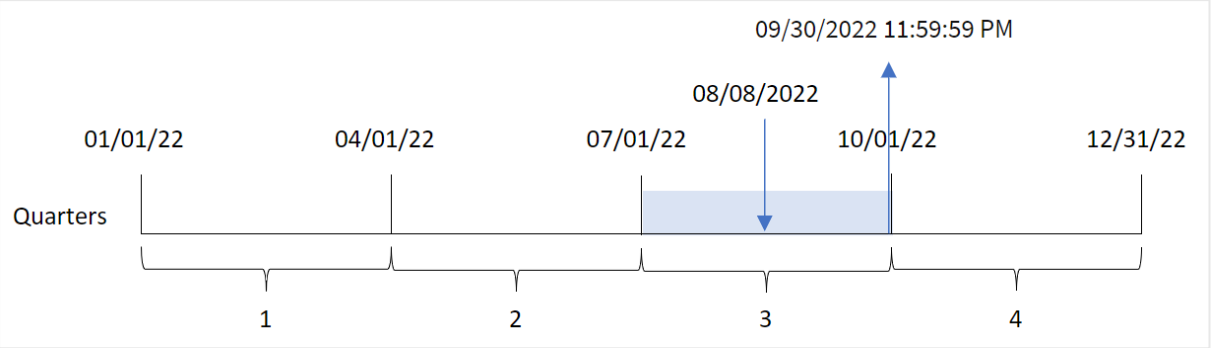
Tabela de resultados

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

O campo "end_of_quarter" é criado na instrução de carregamento anterior usando a função quarterend() e transmitindo o campo de data como o argumento da função.

A função quarterend() identifica inicialmente em qual trimestre o valor da data cai e, em seguida, retorna um carimbo de data/hora para o último milissegundo desse trimestre.

Diagrama da função `quarterend()` com o final do trimestre da transação 8203 identificado



A transação 8203 ocorreu em 8 de agosto. A função `quarterend()` identifica que a transação ocorreu no terceiro trimestre e retorna o último milissegundo desse trimestre, que é 30 de setembro às 11:59:59 PM.

Exemplo 2: period_no

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações em 2022 que é carregado em uma tabela denominada "`Transactions`".
- Um carregamento anterior que contém o seguinte:
  - A função `quarterend()` que é definida como o campo "`previous_quarter_end`" e retorna um carimbo de data/hora para o final do trimestre antes da transação.
  - A função `timestamp()` que é definida como o campo "`previous_end_of_quarter_timestamp`" e retorna o carimbo de data/hora exato do final do trimestre antes da transação.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `id`
- `date`
- `previous_quarter_end`
- `previous_quarter_end_timestamp`

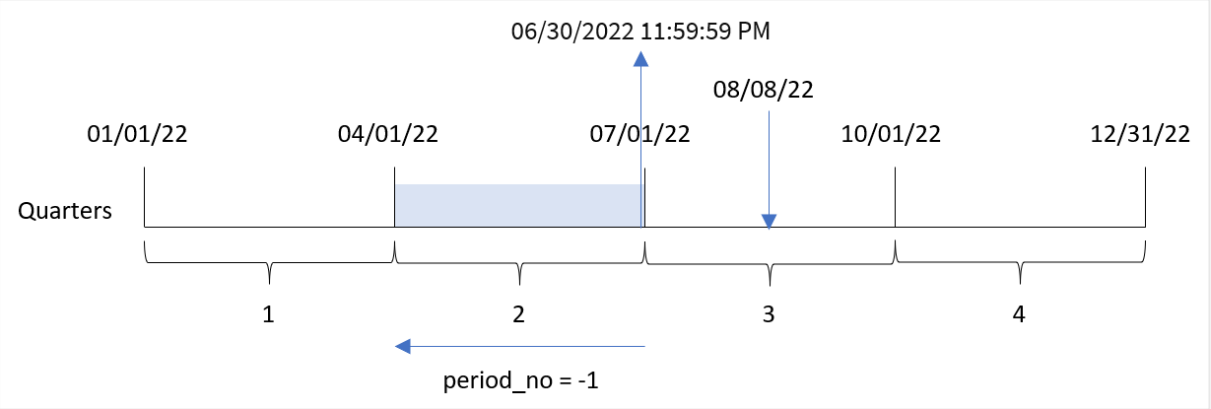
Tabela de resultados

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
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

id	date	previous_quarter_end	previous_quarter_end_timestamp
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

Como um `period_no` de `-1` foi usado como o argumento offset na função `quarterend()`, a função primeiro identifica o ano em que as transações ocorrem. Em seguida, ela muda para um trimestre antes e identifica o último milissegundo desse trimestre.

Diagrama da função `quarterend()` com um period_no de -1



A transação 8203 ocorreu em 8 de agosto. A função `quarterend()` identifica que o trimestre anterior à realização da transação foi entre 1º de abril e 30 de junho. A função então retorna o último milissegundo desse trimestre, 30 de junho, às 11:59:59 PM.

Exemplo 3: first_month_of_year

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações em 2022 que é carregado em uma tabela denominada "`Transactions`".

- Um carregamento anterior que contém o seguinte:
    - A função `quarterend()` que é definida como o campo "`end_of_quarter`" e retorna um carimbo de data/hora para o final do trimestre em que as transações ocorreram.
    - A função `timestamp()` que é definida como o campo "`end_of_quarter_timestamp`" e retorna o carimbo de data/hora exato do final do trimestre selecionado.

No entanto, neste exemplo, a política da empresa é que o exercício financeiro comece em 1º de março.

Script de carregamento

```
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
];
```

Resultados

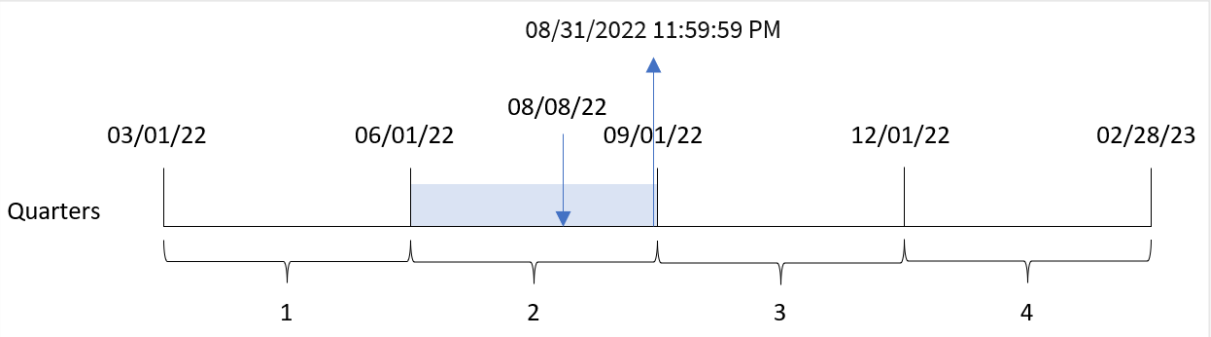
Tabela de resultados

id	date	end_of_quarter	end_of_quarter_timestamp
8188	1/7/2022	02/28/2022	2/28/2022 11:59:59 PM

id	date	end_of_quarter	end_of_quarter_timestamp
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

Como o argumento `first_month_of_year` de `3` é usado na função `quarterend()`, o início do ano passa de 1º de janeiro a 1º de março.

Diagrama da função `quarterend()` com março como o primeiro mês do ano



A transação 8203 ocorreu em 8 de agosto. Como o início do ano é em 1º de março, os trimestres do ano ocorrem entre março e maio, junho e agosto, setembro e novembro e dezembro e fevereiro.

A função quarterend() identifica que a transação ocorreu no trimestre entre o início de junho e agosto e retorna o último milissegundo desse trimestre, que é 31 de agosto às 11:59:59 PM.

Exemplo 4: Exemplo de objeto de gráfico

Script de carregamento e expressão de gráfico

Visão geral

São usados o mesmo conjunto de dados e cenário do primeiro exemplo.

No entanto, neste exemplo, o conjunto de dados permanece inalterado e é carregado no aplicativo. O cálculo que retorna um carimbo de data/hora para o final do trimestre em que as transações ocorreram é criado como uma medida em um gráfico no aplicativo.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- id
- date

Para calcular a data de término do trimestre em que uma transação ocorre, crie as seguintes medidas:

- =quarterend(date)
- =timestamp(quarterend(date))

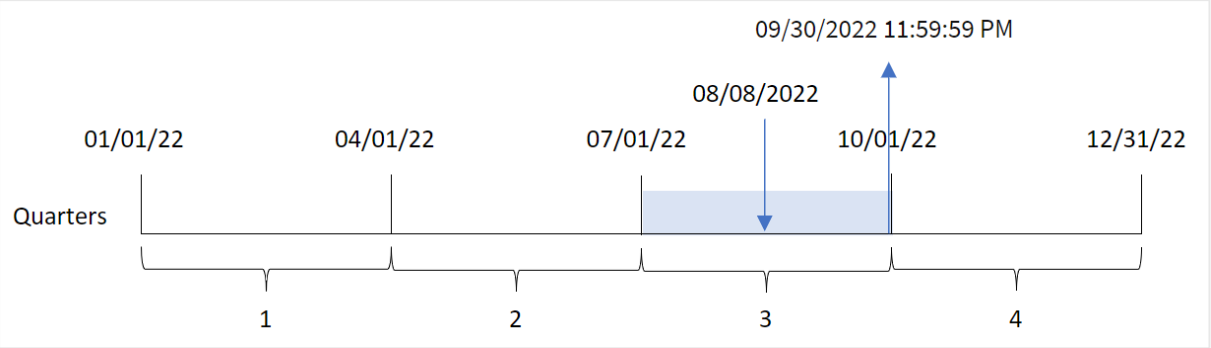
Tabela de resultados

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
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

O campo "end_of_quarter" é criado na instrução de carregamento anterior usando a função quarterend() e transmitindo o campo de data como o argumento da função.

A função quarterend() identifica inicialmente em qual trimestre o valor da data cai e, em seguida, retorna um carimbo de data/hora para o último milissegundo desse trimestre.

Diagrama da função `quarterend()` com o final do trimestre da transação 8203 identificado



A transação 8203 ocorreu em 8 de agosto. A função `quarterend()` identifica que a transação ocorreu no terceiro trimestre e retorna o último milissegundo desse trimestre, que é 30 de setembro às 11:59:59 PM.

Exemplo 5: Cenário

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados é carregado em uma tabela chamada "`Employee_Expenses`". A tabela contém os seguintes campos:
    - IDs de funcionários
    - Nomes de funcionários
    - As reivindicações de despesas médias diárias de cada funcionário.

O usuário final deseja um objeto de gráfico que mostre, por ID de funcionário e nome de funcionário, as reivindicações de despesas estimadas ainda a serem acumuladas para o restante do trimestre. O exercício financeiro começa em janeiro.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `employee_id`
- `employee_name`

Para calcular os juros acumulados, crie a seguinte medida:

- `=(quarterend(today(1))-today(1))*avg_daily_claim`

Defina o **Formato numérico** da medida como **Dinheiro**.

Tabela de resultados

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

A função `quarterend()` usa a data de hoje como seu único argumento e retorna a data de término do mês atual. Em seguida, ela subtrai a data de hoje da data de término do ano, e a expressão retorna o número de dias que restam neste mês.

Esse valor é então multiplicado pela média de solicitações de despesas diárias de cada funcionário para calcular o valor estimado das solicitações que cada funcionário deve fazer no trimestre restante.

quartername

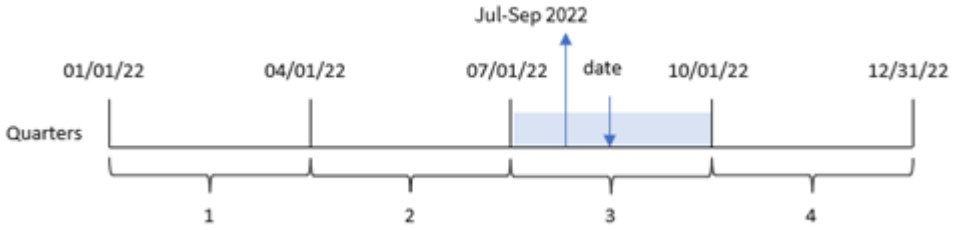
Esta função retorna um valor de exibição que mostra os meses do trimestre (formatados de acordo com a variável de script **MonthNames**) e o ano com um valor numérico subjacente que corresponde a um carimbo de hora do primeiro milissegundo do primeiro dia do trimestre.

Sintaxe:

```
QuarterName(date[, period_no[, first_month_of_year]])
```

Tipo de dados de retorno: dual

Diagrama da função quartername()



A função quartername() determina em qual trimestre a data cai. Em seguida, ela retorna um valor mostrando os meses de início e fim deste trimestre, bem como o ano. O valor numérico subjacente desse resultado é o primeiro milissegundo do trimestre.

Argumentos

Argumento	Descrição
date	A data ou o carimbo de data/hora a ser avaliado.
period_no	**period_no** é um inteiro, em que o valor 0 indica o trimestre que contém **date**. Valores negativos em **period_no** indicam trimestres precedentes e valores positivos indicam trimestres sucessivos.
first_month_of_year	Se desejar trabalhar com anos (fiscais) que não comecem em janeiro, indique um valor entre 2 e 12 em **first_month_of_year**.

Quando usar

A função quartername() é útil quando você deseja comparar agregações por trimestre. Por exemplo, se você quiser ver o total de vendas de produtos por trimestre.

Essa função pode ser usada no script de carregamento para criar um campo em uma tabela do calendário mestre. Como alternativa, ela pode ser usada diretamente em um gráfico como uma dimensão calculada.

Esses exemplos usam o formato de data MM/DD/AAAA. O formato de data é especificado no comando SET DateFormat na parte superior do seu script de carregamento de dados. Altere o formato nos exemplos para atender às suas necessidades.

Exemplos de funções

Exemplo	Resultado
quartername('10/29/2013')	Retorna Oct-Dec 2013.
quartername('10/29/2013', -1)	Retorna Jul-Sep 2013.
quartername('10/29/2013', 0, 3)	Retorna Sep-Nov 2013.

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplo 1: data sem argumentos adicionais

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações para 2022, que é carregado em uma tabela denominada `Transactions`.
- O campo de data fornecido no formato da `variável` de sistema DateFormat (MM/DD/AAAA).
- A criação de um campo, `transaction_quarter`, que retorna o trimestre em que as transações ocorreram.

Adicione seu outro texto aqui, conforme necessário, com listas etc.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `transaction_quarter`

Tabela de resultados

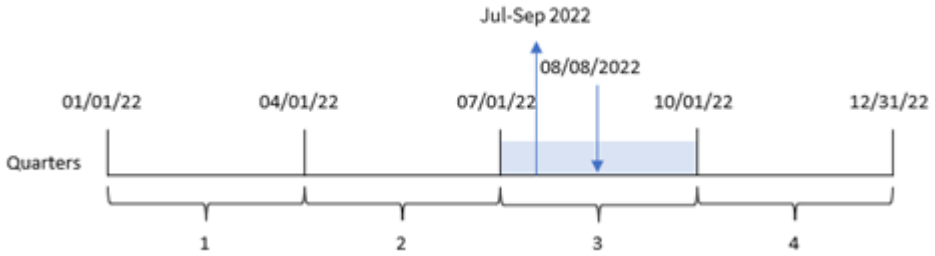
date	transaction_quarter
1/7/2022	Jan-Mar 2022
1/19/2022	Jan-Mar 2022
2/5/2022	Jan-Mar 2022
2/28/2022	Jan-Mar 2022
3/16/2022	Jan-Mar 2022
4/1/2022	Apr-Jun 2022
5/7/2022	Apr-Jun 2022
5/16/2022	Apr-Jun 2022
6/15/2022	Apr-Jun 2022
6/26/2022	Apr-Jun 2022
7/9/2022	Jul-Sep 2022
7/22/2022	Jul-Sep 2022
7/23/2022	Jul-Sep 2022

date	transaction_quarter
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

O campo `transaction_quarter` é criado na instrução de carregamento anterior usando a função `quartername ()` e transmitindo o campo de data como o argumento da função.

A função `quartername()` identifica inicialmente o trimestre no qual o valor da data cai. Em seguida, ela retorna um valor mostrando os meses de início e fim deste trimestre, bem como o ano.

Diagrama da função `quartername()`, exemplo sem argumentos adicionais



A transação 8203 ocorreu em 8 de agosto de 2022. A função `quartername()` identifica que a transação ocorreu no terceiro trimestre e, portanto, retorna de julho a setembro de 2022. Os meses são exibidos no mesmo formato da variável de sistema `MonthNames`.

Exemplo 2: data com o argumento period_no

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- O mesmo conjunto de dados e cenário do primeiro exemplo.
- A criação de um campo, `previous_quarter`, que retorna o trimestre anterior para quando as transações ocorreram.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `previous_quarter`

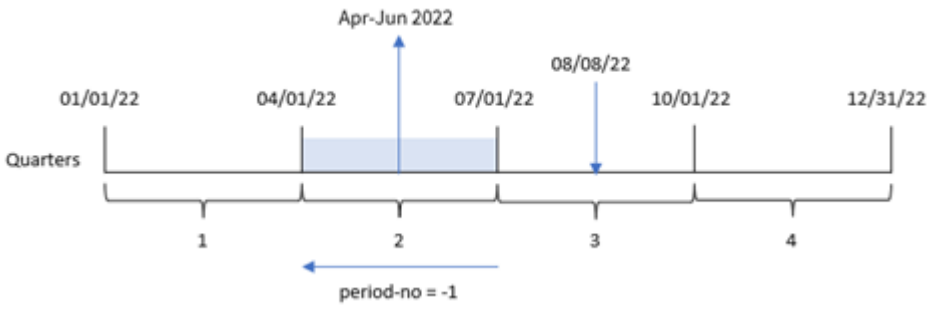
Tabela de resultados

date	previous_quarter
1/7/2022	Oct-Dec 2021
1/19/2022	Oct-Dec 2021
2/5/2022	Oct-Dec 2021

date	previous_quarter
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

Nesse caso, como um `period_no` de -1 foi usado como argumento offset na função `quartername()`, a função primeiro identifica que as transações ocorreram no terceiro trimestre. Em seguida, ele muda para um trimestre antes e retorna um valor que mostra os meses de início e término deste trimestre, bem como o ano.

Diagrama da função `quartername()`, exemplo de period_no



A transação 8203 ocorreu em 8 de agosto. A função `quartername()` identifica que o trimestre anterior à realização da transação foi entre 1º de abril e 30 de junho. Portanto, ela retorna de abril a junho de 2022.

Exemplo 3: data com o argumento first_week_day

Script de carregamento e resultados

Visão geral

Abra o editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém o mesmo conjunto de dados e cenário do primeiro exemplo. No entanto, neste exemplo, precisamos definir 1º de março como o início do exercício financeiro.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

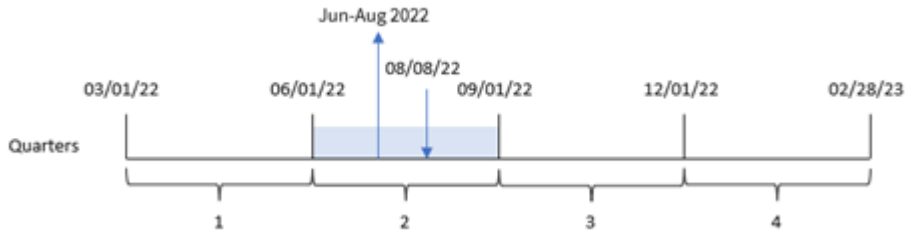
- `date`
- `transaction_quarter`

Tabela de resultados

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
9/26/2022	Sep-Nov 2022
10/14/2022	Sep-Nov 2022
10/29/2022	Sep-Nov 2022

Nesse caso, como o argumento `first_month_of_year` de 3 é usado na função `quartername()`, o início do ano passa de 1º de janeiro a 1º de março. Portanto, os trimestres do ano são separados em março-maio, junho-agosto, setembro-novembro e dezembro-fevereiro.

Diagrama da função `quartername()`, exemplo de first_week_day



A transação 8203 ocorreu em 8 de agosto. A função `quartername()` identifica que a transação ocorreu no segundo trimestre, entre o início de junho e o final de agosto. Portanto, ela retorna Jun-Aug 2022.

Exemplo 4: Exemplo de objeto de gráfico

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém o mesmo conjunto de dados e cenário do primeiro exemplo.

No entanto, neste exemplo, o conjunto de dados inalterado é carregado no aplicativo. O cálculo que retorna um carimbo de data/hora para o final do trimestre em que as transações ocorreram é criado como uma medida em um objeto de gráfico do aplicativo.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão: `date`.

Crie a seguinte medida:

```
=quartername(date)
```

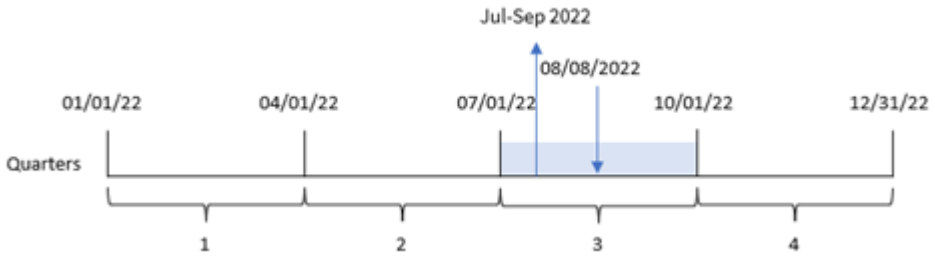
Tabela de resultados

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
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

A medida `transaction_quarter` é criada no objeto de gráfico usando a função `quartername()` e transmitindo o campo `date` como o argumento da função.

A função `quartername()` identifica inicialmente o trimestre no qual o valor da data cai. Em seguida, ela retorna um valor mostrando os meses de início e fim deste trimestre, bem como o ano.

Diagrama da função `quartername()`, exemplo de objeto de gráfico



A transação 8203 ocorreu em 8 de agosto de 2022. A função `quartername()` identifica que a transação ocorreu no terceiro trimestre e, portanto, retorna de julho a setembro de 2022. Os meses são exibidos no mesmo formato da variável de sistema `MonthNames`.

Exemplo 5: Cenário

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações para 2022, que é carregado em uma tabela chamada `Transactions`.
- O campo de data fornecido no formato da variável de sistema `DateFormat` (MM/DD/AAAA).

O usuário final gostaria de um objeto de gráfico que apresentasse o total de vendas por trimestre para as transações. Isso pode ser alcançado mesmo quando essa dimensão não está disponível no modelo de dados. Para isso, use a função `quartername()` como uma dimensão calculada no gráfico.

Script de carregamento

```
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
];
```

Resultados

Faça o seguinte:

1. Carregue os dados e abra uma pasta. Crie uma nova tabela.
2. Crie uma dimensão calculada usando a seguinte expressão:
   `=quartername(date)`
3. Em seguida, calcule o total de vendas usando a seguinte medida de agregação:
   `=sum(amount)`
4. Defina o **Formato numérico** da medida como **Dinheiro**.

Tabela de resultados

=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

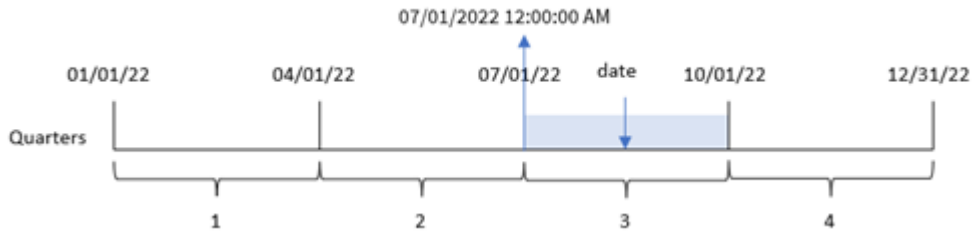
Esta função retorna um valor correspondente a um carimbo de data/hora com o primeiro milissegundo do trimestre que contém **date**. O formato de saída padrão será o **DateFormat** definido no script.

Sintaxe:

```
QuarterStart(date[, period_no[, first_month_of_year]])
```

Tipo de dados de retorno: dual

Diagrama da função quarterstart()



A função quarterstart() determina em qual trimestre a date cai. Em seguida, ela retorna um carimbo de data/hora, no formato de data, para o primeiro milissegundo do primeiro mês desse trimestre.

Argumentos

Argumento	Descrição
date	A data ou o carimbo de data/hora a ser avaliado.
period_no	**period_no** é um inteiro, em que o valor 0 indica o trimestre que contém **date**. Valores negativos em **period_no** indicam trimestres precedentes e valores positivos indicam trimestres sucessivos.
first_month_of_year	Se desejar trabalhar com anos (fiscais) que não comecem em janeiro, indique um valor entre 2 e 12 em **first_month_of_year**.

Quando usar

A função quarterstart() é comumente usada como parte de uma expressão quando o usuário deseja que o cálculo use a fração do trimestre decorrido até o momento. Por exemplo, ela pode ser usada se um usuário quiser calcular os juros acumulados em um trimestre até o momento.

Exemplos de funções

Exemplo	Resultado
`quarterstart('10/29/2005')`	Retorna 10/01/2005.
`quarterstart('10/29/2005', -1 )`	Retorna 07/01/2005.
`quarterstart('10/29/2005', 0, 3)`	Retorna 09/01/2005.

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução SET DateFormat no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplo 1: Sem argumentos adicionais

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações para 2022, que é carregado em uma tabela denominada `Transactions`.
- O campo de data fornecido no formato da `variável` de sistema DateFormat (MM/DD/AAAA).
- A criação de um campo, `start_of_quarter`, que retorna um carimbo de data/hora para o início do trimestre em que as transações ocorreram.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

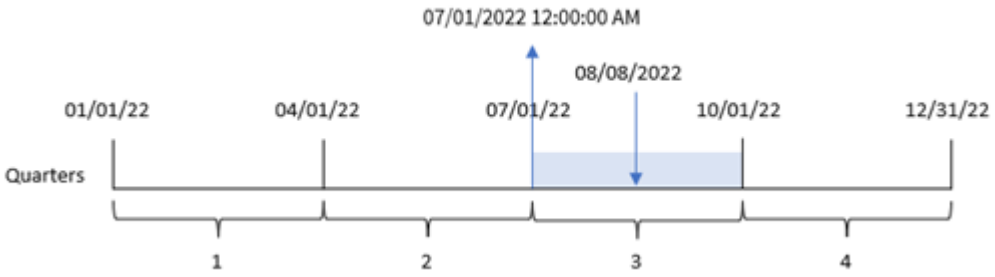
- `date`
- `start_of_quarter`
- `start_of_quarter_timestamp`

Tabela de resultados

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
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

O campo `start_of_quarter` é criado na instrução de carregamento anterior usando a função `quarterstart()` e transmitindo o campo de data como o argumento da função. A função `quarterstart()` identifica inicialmente em qual trimestre o valor da data se enquadra. Em seguida, ela retorna um carimbo de data/hora para o primeiro milissegundo desse trimestre.

Diagrama da função `quarterstart()`, exemplo sem argumentos adicionais



A transação 8203 ocorreu em 8 de agosto. A função `quarterstart()` identifica que a transação ocorreu no terceiro trimestre e retorna o primeiro milissegundo desse trimestre, que é 1º de julho às 12:00:00 AM.

Exemplo 2: period_no

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- O mesmo conjunto de dados e cenário do primeiro exemplo.
- A criação de um campo `previous_quarter_start`, que retorna o carimbo de data/hora do início do trimestre antes da transação.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `previous_quarter_start`
- `previous_quarter_start_timestamp`

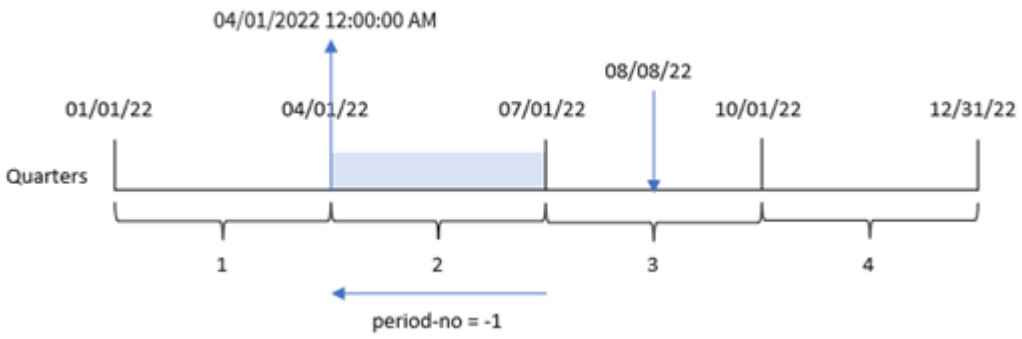
Tabela de resultados

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

date	previous_quarter_start	previous_quarter_start_timestamp
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

Nesse caso, como um `period_no` de -1 foi usado como argumento offset na função `quarterstart()`, a função primeiro identifica o mês em que as transações ocorrem. Em seguida, ela muda para um trimestre antes e identifica o primeiro milissegundo desse trimestre.

Diagrama da função `quarterstart()`, exemplo de period_no



A transação 8203 ocorreu em 8 de agosto. A função `quarterstart()` identifica que o trimestre anterior à realização da transação foi entre 1º de abril e 30 de junho. Em seguida, ela retorna o primeiro milissegundo desse trimestre, 1º de abril, às 12:00:00 AM.

Exemplo 3: first_month_of_year

Script de carregamento e resultados

Visão geral

Abra o editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém o mesmo conjunto de dados e cenário do primeiro exemplo. No entanto, neste exemplo, precisamos definir 1º de março como o início do exercício financeiro.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `start_of_quarter`
- `start_of_quarter_timestamp`

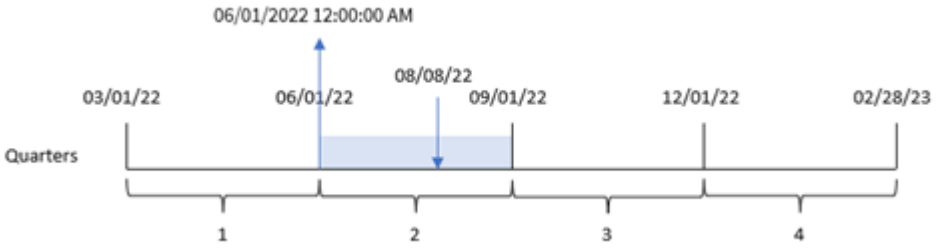
Tabela de resultados

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

date	start_of_quarter	start_of_quarter_timestamp
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
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

Nesse caso, como o argumento `first_month_of_year` de 3 é usado na função `quarterstart()`, o início do ano passa de 1º de janeiro a 1º de março.

Diagrama da função `quarterstart()`, exemplo de first_month_of_year



A transação 8203 ocorreu em 8 de agosto. Como o início do ano é em 1º de março, os trimestres do ano ocorrem entre março e maio, junho e agosto, setembro e novembro e dezembro e fevereiro. A função `quarterstart()` identifica que a transação ocorreu no trimestre entre o início de junho e agosto e retorna o primeiro milissegundo desse trimestre, que é 1º de junho às 12:00:00 AM.

Exemplo 4: Exemplo de objeto de gráfico

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém o mesmo conjunto de dados e cenário do primeiro exemplo.

No entanto, neste exemplo, o conjunto de dados inalterado é carregado no aplicativo. O cálculo que retorna um carimbo de data/hora para o final do trimestre em que as transações ocorreram é criado como uma medida em um objeto de gráfico do aplicativo.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão: `date`.

Adicione as seguintes medidas:

- `=quarterstart(date)`
- `=timestamp(quarterstart(date))`

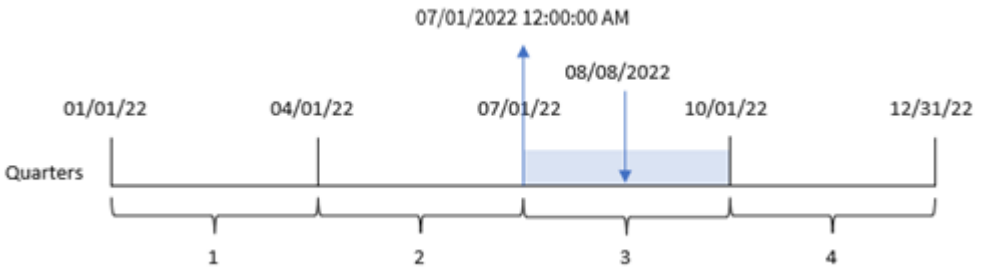
Tabela de resultados

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
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

A medida `start_of_quarter` é criada no objeto de gráfico usando a função `quarterstart()` e transmitindo o campo `date` como o argumento da função.

A função `quarterstart()` identifica o trimestre no qual o valor da data cai, retornando um carimbo de data/hora para o primeiro milissegundo desse trimestre.

Diagrama da função `quarterstart()`, exemplo de objeto de gráfico



A transação 8203 ocorreu em 8 de agosto. A função `quarterstart()` identifica que a transação ocorreu no terceiro trimestre e retorna o primeiro milissegundo desse trimestre. Esse valor retornado é 1º de julho às 12:00:00 AM.

Exemplo 5: Cenário

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de saldos de empréstimos, que é carregado em uma tabela denominada `Loans`.
- Dados que consistem em IDs de empréstimos, saldo no início do trimestre e taxa de juros simples cobrada em cada empréstimo por ano.

O usuário final deseja um objeto de gráfico que mostre, por ID de empréstimo, os juros atuais que foram acumulados em cada empréstimo no trimestre até o momento.

Script de carregamento

```
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
];
```

Resultados

Faça o seguinte:

1. Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:
   - `loan_id`
   - `start_balance`
2. Em seguida, crie essa medida para calcular os juros acumulados:
   `=start_balance*(rate*(today(1)-quarterstart(today(1)))/365)`
3. Defina o **Formato numérico** da medida como **Dinheiro**.

Tabela de resultados

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

A função `quarterstart()`, usando a data de hoje como seu único argumento, retorna a data de início do ano atual. Ao subtrair esse resultado da data atual, a expressão retorna o número de dias decorridos até o momento neste trimestre.

Esse valor é então multiplicado pela taxa de juros e dividido por 365 para retornar a taxa de juros efetiva acumulada do período. O resultado é então multiplicado pelo saldo inicial do empréstimo para retornar os juros acumulados até o momento nesse trimestre.

second

Esta função retorna um número inteiro que representa o segundo em que a fração da **expression** é interpretada como uma hora, de acordo com a interpretação numérica padrão.

Sintaxe:

```
second (expression)
```

Tipo de dados de retorno: inteiro

Quando usar

A função `second()` é útil quando você deseja comparar agregações por segundo. Por exemplo, ela pode ser usada se você deseja ver a distribuição da contagem de atividades por segundo.

Essas dimensões podem ser criadas no script de carregamento usando a função para criar um campo em uma tabela de Calendário mestre ou usadas diretamente em um gráfico como uma dimensão calculada.

Exemplos de funções

Exemplo	Resultado
`second( '09:14:36' )`	retorna 36
`second( '0.5555' )`	retorna 55 (Porque 0.5555 = 13:19:55)

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplo 1: variável

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo transações por carimbo de data/hora que é carregado em uma tabela denominada `Transactions`.
- A `variável` de sistema TimeStamp padrão (`M/D/YYYY h:mm:ss[.fff] TT`) é usada.
- A criação de um campo, `second`, para calcular quando as compras ocorreram.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- date
- second

Tabela de resultados

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

Os valores no campo second são criados usando a função second() e transferindo a data como a expressão no comando de carregamento anterior.

Exemplo 2: Objeto de gráfico

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém o mesmo conjunto de dados e cenário do primeiro exemplo. No entanto, neste exemplo, o conjunto de dados inalterado é carregado no aplicativo. Os valores de second são calculados por meio de uma medida em um objeto de gráfico.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão:date.

Crie a seguinte medida:

```
=second(date)
```

Tabela de resultados

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

Os valores para second são criados usando a função second() e transmitindo a data como a expressão em uma medida para o objeto de gráfico.

Exemplo 3: Cenário

Script de carregamento e expressões de gráfico

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados de carimbos de data/hora, que é gerado para representar o tráfego para o site de venda de ingressos para um festival específico. Esses carimbos de data/hora e um id correspondente são carregados em uma tabela denominada Web_Traffic.
- A variável de sistema TimeStamp M/D/YYYY h:mm:ss[.fff] TT é usada.

Nesse cenário, havia 10.000 ingressos, que foram colocados à venda às 9h do dia 20 de maio de 2021. Um minuto depois, os ingressos estavam esgotados.

O usuário deseja um objeto de gráfico que mostrasse a contagem de visitas ao site por segundo.

Script de carregamento

```
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;
```

Resultados

Faça o seguinte:

1. Carregue os dados e abra uma pasta. Crie uma nova tabela.
2. Em seguida, crie uma dimensão calculada usando a seguinte expressão:

```
=second(timestamp)
```

3. Crie uma medida de agregação para calcular a contagem total de entradas:
```
=count(id)
```

A tabela de resultados será semelhante à tabela abaixo, mas com valores diferentes para a medida de agregação:

Tabela de resultados

second(timestamp)	=count(id)
0	150
1	184
2	163
3	178
4	179
5	158
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
Mais de 39 linhas	

setdateyear

Esta função admite como entrada um **timestamp** e **year** e atualiza o **timestamp** com o **year** especificado na entrada.

Sintaxe:

```
setdateyear (timestamp, year)
```

Tipo de dados de retorno: dual

Argumentos:

Argumentos

Argumento	Descrição
timestamp	Um carimbo de data/hora padrão do Qlik Sense (muitas vezes, apenas uma data).
year	Um ano com quatro dígitos.

Exemplos e resultados:
Estes exemplos usam o formato de data **DD/MM/YYYY**. O formato de data é especificado no comando **SET DateFormat** na parte superior do seu script de carga de dados. Altere o formato nos exemplos para atender às suas necessidades.

Exemplos de scripts

Exemplo	Resultado
`setdateyear ('29/10/2005', 2013)`	Retorna '29/10/2013
`setdateyear ('29/10/2005 04:26:14', 2013)`	Retorna '29/10/2013 04:26:14' Para ver a hora parte do carimbo de data/hora em uma visualização, você deve definir a formatação numérica da Data e escolher um valor para Formatação que exiba os valores de tempo.

Exemplo:

Adicione o script de exemplo ao seu aplicativo e execute-o. Para ver o resultado, adicione os campos listados na coluna de resultados a uma pasta no seu aplicativo.

```
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

];
```

A tabela resultante contém as datas originais e uma coluna na qual a data foi acertada para 2013.

Tabela de resultados

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

Esta função admite como entrada um **timestamp**, um **month** e um **year** , e atualiza o **timestamp** com o **year** e o **month** especificado na entrada. .

Sintaxe:

```
SetDateYearMonth (timestamp, year, month)
```

Tipo de dados de retorno: dual

Argumentos:

Argumentos

Argumento	Descrição
timestamp	Um carimbo de data/hora padrão do Qlik Sense (muitas vezes, apenas uma data).
year	Um ano com quatro dígitos.
month	Um mês com um ou dois dígitos.

Exemplos e resultados:
Estes exemplos usam o formato de data **DD/MM/YYYY**. O formato de data é especificado no comando **SET DateFormat** na parte superior do seu script de carga de dados. Altere o formato nos exemplos para atender às suas necessidades.

Exemplos de scripts

Exemplo	Resultado
`setdateyearmonth ('29/10/2005', 2013, 3)`	Retorna '29/03/2013
`setdateyearmonth ('29/10/2005 04:26:14', 2013, 3)`	Retorna '29/03/2013 04:26:14' Para ver a hora parte do carimbo de data/hora em uma visualização, você deve definir a formatação numérica da Data e escolher um valor para Formatação que exiba os valores de tempo.

Exemplo:

Adicione o script de exemplo ao seu aplicativo e execute-o. Para ver o resultado, adicione os campos listados na coluna de resultados a uma pasta no seu aplicativo.

```
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

];
```

A tabela resultante contém as datas originais e uma coluna na qual a data foi acertada para 2013.

Tabela de resultados

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

Esta função retorna o fuso horário, conforme definido no computador onde o mecanismo do Qlik está executando.

Sintaxe:

```
TimeZone( )
```

Tipo de dados de retorno: dual

Exemplo:

`timezone( )`
Se quiser ver um fuso horário diferente em uma medida em seu aplicativo, você pode usar a função `localtime()` em uma medida.

today

Essa função retorna a data atual. A função retorna valores no formato da variável do sistema `DateFormat`.

Sintaxe:

```
today([ timer_mode])
```

Tipo de dados de retorno: dual

A função `today()` pode ser usada no script de carregamento ou em objetos de gráfico.

O valor padrão de `timer_mode` é 1.

Argumentos

Argumento	Descrição
timer_mode	Pode ter os seguintes valores: 0 (dia do último carregamento de dados finalizado) 1 (dia da chamada de função) 2 (dia em que o aplicativo foi aberto) *Se você usar a função em um script de carregamento de dados, **timer_mode=0** resultará no dia do último carregamento de dados finalizado, enquanto **timer_mode=1** indicará o dia do carregamento de dados atual.*

Exemplos de funções

timer_mode value	Resultado se usado no script de carregamento	Resultado se usado no objeto de gráfico
0	Retorna uma data, no formato da `variável` de sistema DateFormat, da última recarga de dados bem-sucedida antes do último carregamento de dados.	Retorna uma data, no formato da variável de sistema `DateFormat`, para o último carregamento de dados.

timer_mode value	Resultado se usado no script de carregamento	Resultado se usado no objeto de gráfico
1	Retorna uma data, no formato da variável de sistema `DateFormat`, para o último carregamento de dados.	Retorna uma data, no formato da variável de sistema `DateFormat`, da chamada da função.
2	Retorna uma data, no formato de variável de sistema `DateFormat`, de quando a sessão do usuário no aplicativo começou. Isso não será atualizado a menos que o usuário recarregue o script.	Retorna a data, no formato de variável de sistema `DateFormat`, de quando a sessão do usuário no aplicativo começou. Isso será atualizado quando uma nova sessão for iniciada ou quando os dados do aplicativo forem recarregados.

Quando usar

A função `today()` é normalmente usada como um componente dentro de uma expressão. Por exemplo, ela pode ser usada para calcular os juros acumulados em um mês até a data atual.

A tabela a seguir fornece uma explicação do resultado retornado pela função `today()`, considerando valores diferentes para o argumento `timer_mode`:

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplo 1: geração de objetos usando script de carregamento

Script de carregamento e resultados

Visão geral

O exemplo a seguir cria três variáveis usando a função `today()`. Cada variável usa uma das opções `timer_mode` para demonstrar seu efeito.

Para que as variáveis demonstrem sua finalidade, recarregue o script e, depois de 24 horas, recarregue o script uma segunda vez. Isso fará com que as variáveis `today(0)` e `today(1)` mostrem valores diferentes, demonstrando corretamente sua finalidade.

Script de carregamento

```
LET vPreviousDataLoad = today(0);
LET vCurrentDataLoad = today(1);
LET vApplicationOpened = today(2);
```

Resultados

Depois que os dados tiverem sido carregados pela segunda vez, crie três caixas de texto usando as instruções abaixo.

Primeiro, crie uma caixa de texto para os dados que foram carregados anteriormente.

Faça o seguinte:

1. Usando o objeto de gráfico **Texto e imagem**, crie uma caixa de texto.
2. Adicione a seguinte medida ao objeto:
   `=vPreviousDataLoad`
3. Em **Aparência**, selecione **Show titles** e adicione o título "Tempo de carregamento anterior" ao objeto.

Em seguida, crie uma caixa de texto para os dados que estão sendo carregados.

Faça o seguinte:

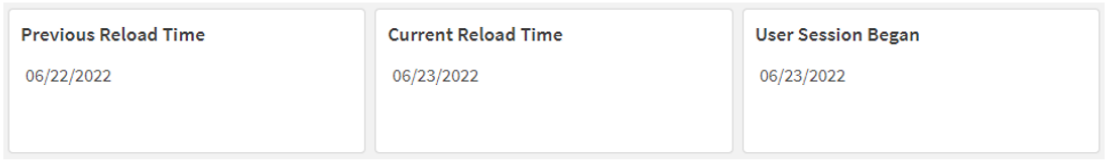
1. Usando o objeto de gráfico **Texto e imagem**, crie uma caixa de texto.
2. Adicione a seguinte medida ao objeto:
   `=vCurrentDataLoad`
3. Em **Aparência**, selecione **Show titles** e adicione o título "Tempo de carregamento atual" ao objeto.

Crie uma caixa de texto final para mostrar quando a sessão do usuário no aplicativo foi iniciada.

Faça o seguinte:

1. Usando o objeto de gráfico **Texto e imagem**, crie uma caixa de texto.
2. Adicione a seguinte medida ao objeto:
   `=vApplicationOpened`
3. Em **Aparência**, selecione **Show titles** e adicione o título "Sessão do usuário iniciada" ao objeto.

Diagrama de variáveis criadas usando a função `today()` no script de carregamento



A imagem acima mostra exemplos de valores para cada uma das variáveis criadas. Por exemplo, os valores podem ser os seguintes:

- Hora de carregamento anterior: 22/06/2022
- Hora de carregamento atual: 23/06/2022
- Início da sessão do usuário: 23/06/2022

Exemplo 2: geração de objetos sem um script de carregamento

Script de carregamento e expressão de gráfico

Visão geral

O exemplo a seguir cria três objetos de gráfico usando a função `today()`. Cada objeto de gráfico usa uma das opções `timer_mode` para demonstrar seu efeito.

Não há script de carregamento para esse exemplo.

Resultados

Depois que os dados tiverem sido carregados pela segunda vez, crie três caixas de texto.

Primeiro, crie uma caixa de texto para o último carregamento de dados.

Faça o seguinte:

1. Usando o objeto de gráfico **Texto e imagem**, crie uma caixa de texto.
2. Adicione a seguinte medida.
   `=today(0)`
3. Em **Aparência**, selecione **Mostrar títulos** e adicione o título "Carregamento de dados mais recente" ao objeto.

Em seguida, crie uma caixa de texto para mostrar a hora atual.

Faça o seguinte:

1. Usando o objeto de gráfico **Texto e imagem**, crie uma caixa de texto.
2. Adicione a seguinte medida:

```
=today(1)
```

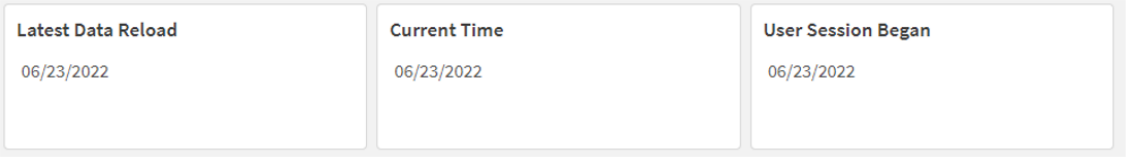
3. Em **Aparência**, selecione **Mostrar títulos** e adicione o título "Hora atual" ao objeto.

Crie uma caixa de texto final para mostrar quando a sessão do usuário no aplicativo foi iniciada.

Faça o seguinte:

1. Usando o objeto de gráfico **Texto e imagem**, crie uma caixa de texto.
2. Adicione a seguinte medida:
   `=today(2)`
3. Em **Aparência**, selecione **Mostrar títulos** e adicione o título “Início da sessão do usuário” ao objeto.

Diagrama de objetos criados usando a função `today()` sem script de carregamento



A imagem acima mostra exemplos de valores para cada um dos objetos criados. Por exemplo, os valores podem ser os seguintes:

- Carregamento de dados mais recente: 23/06/2022
- Hora atual: 23/06/2022
- Início da sessão do usuário: 23/06/2022

O objeto de gráfico “Carregamento de dados mais recente” usa um valor `timer_mode` de 0. Isso retorna o carimbo de data/hora da última vez em que os dados foram carregados com êxito.

O objeto de gráfico “Hora atual” usa um valor `timer_mode` de 1. Isso retorna a hora atual de acordo com o relógio do sistema. Se a pasta ou o objeto forem atualizados, esse valor será atualizado.

O objeto de gráfico “Início da sessão do usuário” usa um valor `timer_mode` de 2. Isso retorna o carimbo de data/hora de quando o aplicativo foi aberto e a sessão do usuário foi iniciada.

Exemplo 3: Cenário

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de saldos de empréstimos, que é carregado em uma tabela denominada `Loans`.
- Tabela de dados com campos para ID de empréstimo, saldo no início do mês e taxa de juros simples cobrada em cada empréstimo por ano.

O usuário final deseja um objeto de gráfico que mostre, por ID de empréstimo, os juros atuais que foram acumulados em cada empréstimo no mês até o momento. Embora o aplicativo seja recarregado apenas uma vez por semana, o usuário gostaria que os resultados fossem atualizados sempre que o objeto ou aplicativo fosse atualizado.

Script de carregamento

```
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
];
```

Resultados

Faça o seguinte:

1. Carregue os dados e abra uma pasta. Crie uma nova tabela.
2. Adicione os seguintes campos como dimensões.
    - `loan_id`
    - `start_balance`
3. Em seguida, crie uma medida para calcular os juros acumulados:
   `=start_balance*(rate*(today(1)-monthstart(today(1)))/365)`
4. Defina o **Formato numérico** da medida como **Dinheiro**.

Tabela de resultados

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

A função monthstart(), usando a função today() para retornar a data de hoje como seu único argumento, retorna a data de início do mês atual. Ao subtrair esse resultado da data atual, novamente usando a função today(), a expressão retorna o número de dias decorridos até o momento neste mês.

Esse valor é então multiplicado pela taxa de juros e dividido por 365 para retornar a taxa de juros efetiva acumulada desse período. O resultado é então multiplicado pelo saldo inicial do empréstimo para retornar os juros acumulados até o momento neste mês.

Como o valor de 1 é usado como o argumento timer_mode nas funções today() dentro da expressão, todas as vezes que o objeto de gráfico é atualizado (abrindo o aplicativo, atualizando a página, movendo-se entre pastas etc.), a data retornada será referente à data atual, e os resultados serão atualizados de acordo.

UTC

Retorna o atual Coordinated Universal Time.

Sintaxe:

```
UTC( )
```

Tipo de dados de retorno: dual

Exemplo:

```
utc( )
```

week

Essa função retorna um número inteiro representando o número da semana correspondente à data inserida.

Sintaxe:

```
week(timestamp [, first_week_day [, broken_weeks [, reference_day]]])
```

Tipo de dados de retorno: inteiro

Argumentos

Argumento	Descrição
timestamp	A data ou o carimbo de data/hora a ser avaliado.
first_week_day	Especifica o dia no qual inicia a semana. Se omitido, o valor da variável **FirstWeekDay** será usado. Os valores possíveis **first_week_day** são 0 para segunda-feira, 1 para terça, 2 para quarta-feira, 3 para quinta-feira, 4 para sexta-feira, 5 para sábado e 6 para domingo. Para obter mais informações sobre a variável de sistema, consulte *FirstWeekDay (page 226)*.

Argumento	Descrição
broken_ weeks	Se você não especificar **broken_weeks**, o valor da variável **BrokenWeeks** será usado para definir se as semanas são quebradas ou não.
reference_ day	Se você não especificar **reference_day**, o valor da variável **ReferenceDay** será usado para definir qual dia em janeiro será definido como dia de referência para definir a semana 1. Por padrão, as funções Qlik Sense usam 4 como o dia de referência. Isso significa que a semana 1 deve conter 4 de janeiro, ou dito de outra forma, que a semana 1 deve sempre ter pelo menos 4 dias em janeiro.

A função `week()` determina em qual semana a data cai e retorna o número da semana.

No Qlik Sense, as configurações regionais são obtidas quando o aplicativo é criado, e as configurações correspondentes são armazenadas no script como variáveis de ambiente. Elas são usados para determinar o número da semana.

Isso significa que a maioria dos desenvolvedores de aplicativos europeus obtém as seguintes variáveis de ambiente, correspondentes à definição ISO 8601:

```
Set FirstWeekDay =0;    // Monday as first week day
Set BrokenWeeks  =0;    // Use unbroken weeks
Set ReferenceDay =4;    // Jan 4th is always in week 1
```

Um desenvolvedor de aplicativos norte-americano geralmente recebe as seguintes variáveis de ambiente:

```
Set FirstWeekDay =6;    // Sunday as first week day
Set BrokenWeeks  =1;    // Use broken weeks
Set ReferenceDay =1;    // Jan 1st is always in week 1
```

O primeiro dia da semana é determinado pela variável de sistema `FirstWeekDay`. Você também pode alterar o primeiro dia da semana usando o argumento `first_week_day` na função `week()`.

Se seu aplicativo usar semanas interrompidas, a contagem de números de semana começará em 1º de janeiro e terminará no dia anterior à variável do sistema `FirstWeekDay`, independentemente de quantos dias tenham ocorrido.

Se o seu aplicativo estiver usando semanas não quebradas, a semana 1 pode começar no ano anterior ou nos primeiros dias de janeiro. Isso depende de como você usa as variáveis de ambiente `FirstWeekDay` e `ReferenceDay`.

Quando usar

A função `The week()` é útil quando você deseja comparar agregações por semanas. Por exemplo, ela pode ser usada se você deseja ver o total de vendas de produtos por semana. A função `week()` é escolhida no lugar de `weekname()` quando o usuário deseja que o cálculo não use necessariamente as variáveis de sistema `BrokenWeeks`, `FirstWeekDay` ou `ReferenceDay` do aplicativo.

Por exemplo, se você quiser ver o total de vendas de produtos por semana.

Se o aplicativo estiver usando semanas não quebradas, a semana 1 pode conter datas de dezembro do ano anterior ou pode excluir datas em janeiro do ano atual. Se o aplicativo estiver usando semanas quebradas, a semana 1 pode conter menos de sete dias.

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Os exemplos abaixo pressupõem

```
Set DateFormat= 'MM/DD/YYYY';
Set FirstWeekDay=0;
Set BrokenWeeks=0;
Set ReferenceDay=4;
```

Exemplos de funções

Exemplo	Resultado
`week('12/28/2021')`	Retorna 52.
`week(44614)`	Retorna 8, já que esse é o número de série de 22/02/2022.
`week('01/03/2021')`	Retorna 53.
`week('01/03/2021',6)`	Returna 1.

Exemplo 1: Variáveis padrão do sistema

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações da última semana de 2021 e das primeiras duas semanas de 2022. Esse conjunto é carregado em uma tabela denominada `Transactions`.
- O campo de data fornecido no formato da variável de sistema `DateFormat` (MM/DD/AAAA).
- A criação de um campo, `week_number`, que retorna o número do ano e da semana em que as transações ocorreram.

- A criação de um campo chamado week_day, mostrando o valor do dia da semana de cada data da transação.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- id
- date
- week_day
- week_number

Tabela de resultados

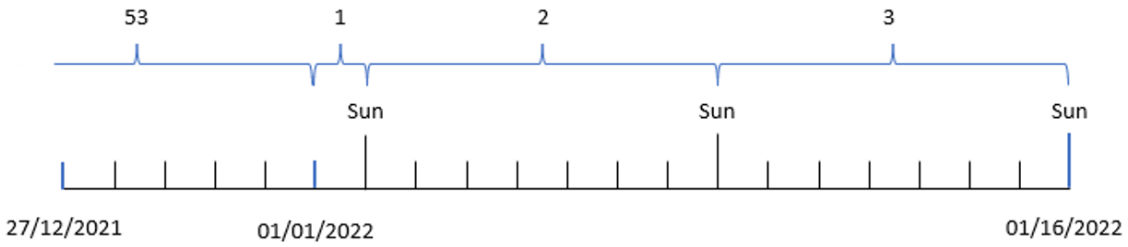
id	date	week_day	week_number
8183	12/27/2021	Seg	53
8184	12/28/2021	Ter	53
8185	12/29/2021	Qua	53
8186	12/30/2021	Qui	53
8187	12/31/2021	Sex	53
8188	01/01/2022	Sáb	1
8189	01/02/2022	Dom	2
8190	01/03/2022	Seg	2
8191	01/04/2022	Ter	2
8192	01/05/2022	Qua	2
8193	01/06/2022	Qui	2
8194	01/07/2022	Sex	2
8195	01/08/2022	Sáb	2
8196	01/09/2022	Dom	3
8197	01/10/2022	Seg	3
8198	01/11/2022	Ter	3
8199	01/12/2022	Qua	3
8200	01/13/2022	Qui	3
8201	01/14/2022	Sex	3

O campo `week_number` é criado na instrução de carregamento anterior usando a função `week()` e transmitindo o campo `date` como o argumento da função.

Nenhum outro parâmetro é informado para a função e, portanto, as seguintes variáveis padrão que afetam a função `week()` estão em vigor:

- `BrokenWeeks`: A contagem semanal começa em 1º de janeiro
- `FirstWeekDay`: O primeiro dia da semana é domingo

Diagrama da função week(), usando variáveis de sistema padrão



Como o aplicativo está usando a variável de sistema `BrokenWeeks` padrão, a semana 1 começa em 1º de janeiro, um sábado.

Por causa da variável de sistema `FirstWeekDay` padrão, as semanas começam em um domingo. O primeiro domingo após 1º de janeiro ocorre em 2 de janeiro, que é quando começa a semana 2.

Exemplo 2: first_week_day

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- A criação de um campo, `week_number`, que retorna o número do ano e da semana em que as transações ocorreram.
- A criação de um campo chamado `week_day`, mostrando o valor do dia da semana de cada data da transação.

Neste exemplo, gostaríamos de definir o início da semana de trabalho como terça-feira.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `id`
- `date`
- `week_day`
- `week_number`

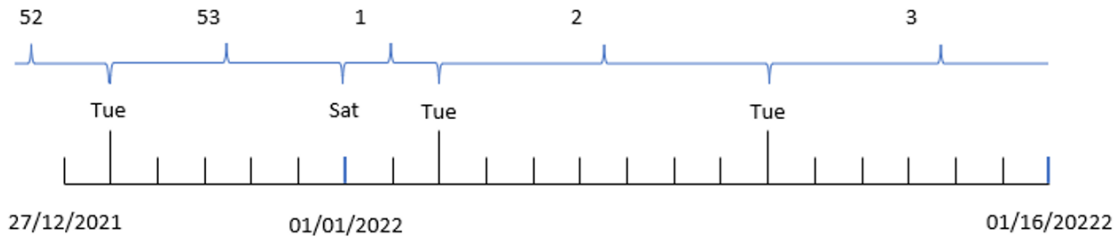
Tabela de resultados

id	date	week_day	week_number
8183	12/27/2021	Seg	52
8184	12/28/2021	Ter	53
8185	12/29/2021	Qua	53
8186	12/30/2021	Qui	53
8187	12/31/2021	Sex	53
8188	01/01/2022	Sáb	1
8189	01/02/2022	Dom	1
8190	01/03/2022	Seg	1
8191	01/04/2022	Ter	2
8192	01/05/2022	Qua	2

id	date	week_day	week_number
8193	01/06/2022	Qui	2
8194	01/07/2022	Sex	2
8195	01/08/2022	Sáb	2
8196	01/09/2022	Dom	2
8197	01/10/2022	Seg	2
8198	01/11/2022	Ter	3
8199	01/12/2022	Qua	3
8200	01/13/2022	Qui	3
8201	01/14/2022	Sex	3

O aplicativo ainda está usando semanas interrompidas. No entanto, o argumento `first_week_day` foi definido como 1 na função `week()`. Isso define o primeiro dia da semana como uma terça-feira.

Diagrama da função week(), exemplo de first_week_day



O aplicativo está usando a variável de sistema `BrokenWeeks` padrão e, portanto, a semana 1 começa em 1º de janeiro, um sábado.

O argumento `first_week_day` da função `week()` define o primeiro dia da semana como uma terça-feira. Portanto, a semana 53 começa em 28 de dezembro de 2021.

No entanto, como a função ainda está usando semanas interrompidas, a semana 1 durará apenas dois dias, pois a primeira terça-feira após 1º de janeiro ocorre em 3 de janeiro.

Exemplo 3: unbroken_weeks

Script de carregamento e resultados

Visão geral

Abra o editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém o mesmo conjunto de dados e cenário do primeiro exemplo.

Neste exemplo, usamos semanas ininterruptas.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `id`
- `date`
- `week_day`
- `week_number`

Diagrama da função week(), exemplo de objeto de gráfico

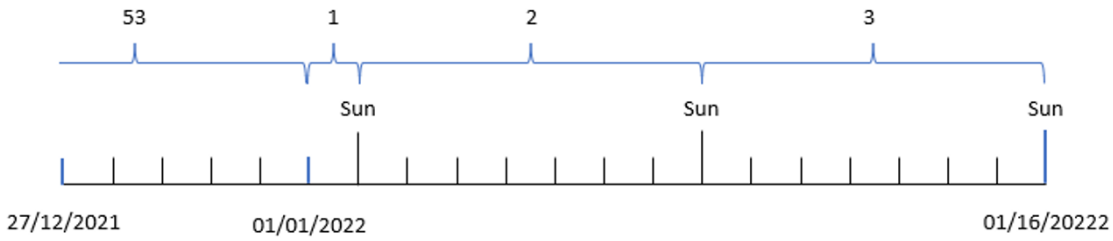


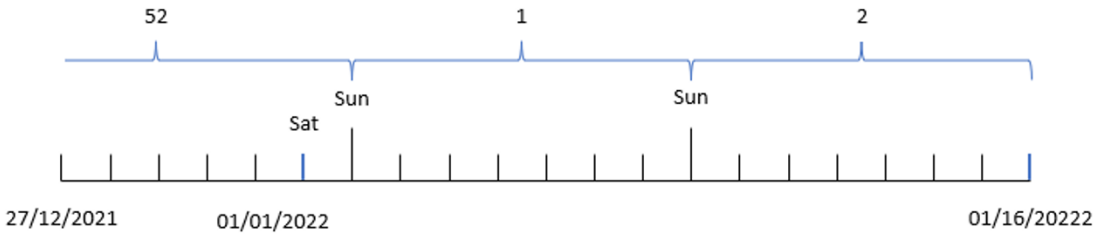
Tabela de resultados

id	date	week_day	week_number
8183	12/27/2021	Seg	52
8184	12/28/2021	Ter	52
8185	12/29/2021	Qua	52
8186	12/30/2021	Qui	52
8187	12/31/2021	Sex	52
8188	01/01/2022	Sáb	52
8189	01/02/2022	Dom	1
8190	01/03/2022	Seg	1
8191	01/04/2022	Ter	1
8192	01/05/2022	Qua	1
8193	01/06/2022	Qui	1
8194	01/07/2022	Sex	1
8195	01/08/2022	Sáb	1
8196	01/09/2022	Dom	2
8197	01/10/2022	Seg	2
8198	01/11/2022	Ter	2
8199	01/12/2022	Qua	2
8200	01/13/2022	Qui	2
8201	01/14/2022	Sex	2

O parâmetro first_week_date está definido como 1, tornando terça-feira o primeiro dia da semana. O parâmetro broken_weeks está definido como 0, forçando a função a usar semanas ininterruptas. Por fim, o terceiro parâmetro define reference_day como 2.

O parâmetro `first_week_date` está definido como 6, tornando o domingo o primeiro dia da semana. O parâmetro `broken_weeks` está definido como 0, forçando a função a usar semanas ininterruptas.

Diagrama da função week(), exemplo de uso de semanas ininterruptas



Ao usar semanas ininterruptas, a semana 1 não começa necessariamente em 1º de janeiro. Em vez disso, ela deve ter no mínimo quatro dias. Portanto, no conjunto de dados, a semana 52 termina no sábado, 1º de janeiro de 2022. A semana 1 então começa na variável de sistema `FirstWeekDay`, que é domingo, 2 de janeiro. Essa semana terminará no sábado seguinte, 8 de janeiro.

Exemplo 4: reference_day

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- O mesmo conjunto de dados e cenário do terceiro exemplo.
- A criação de um campo, `week_number`, que retorna o número do ano e da semana em que as transações ocorreram.
- A criação de um campo chamado `week_day`, mostrando o valor do dia da semana de cada data da transação.

Além disso, as seguintes condições devem ser atendidas:

- A semana de trabalho começa na terça-feira.
- A empresa usa semanas ininterruptas.
- O valor de `reference_day` é 2. Em outras palavras, o número mínimo de dias em janeiro na semana 1 será 2.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- id
- date
- week_day
- week_number

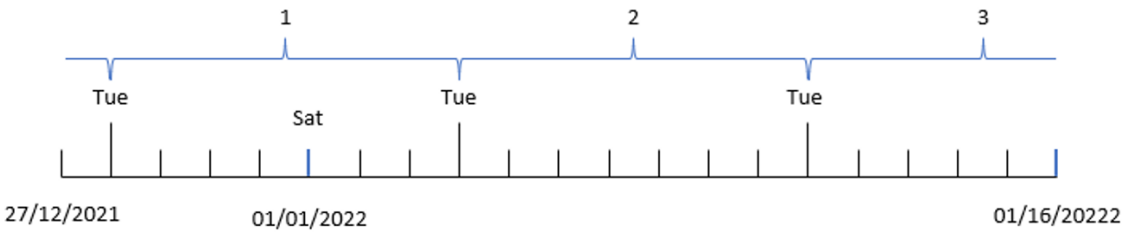
Tabela de resultados

id	date	week_day	week_number
8183	12/27/2021	Seg	52
8184	12/28/2021	Ter	1
8185	12/29/2021	Qua	1
8186	12/30/2021	Qui	1

id	date	week_day	week_number
8187	12/31/2021	Sex	1
8188	01/01/2022	Sáb	1
8189	01/02/2022	Dom	1
8190	01/03/2022	Seg	1
8191	01/04/2022	Ter	2
8192	01/05/2022	Qua	2
8193	01/06/2022	Qui	2
8194	01/07/2022	Sex	2
8195	01/08/2022	Sáb	2
8196	01/09/2022	Dom	2
8197	01/10/2022	Seg	2
8198	01/11/2022	Ter	3
8199	01/12/2022	Qua	3
8200	01/13/2022	Qui	3
8201	01/14/2022	Sex	3

O parâmetro `first_week_date` está definido como 1, tornando terça-feira o primeiro dia da semana. O parâmetro `broken_weeks` está definido como 0, forçando a função a usar semanas ininterruptas. Por fim, o terceiro parâmetro define o parâmetro `reference_day` como 2.

Diagrama da função week(), exemplo de reference_day



Com a função usando semanas ininterruptas e um valor `reference_day` de 2 usado como parâmetro, a semana 1 só precisa incluir dois dias em janeiro. Como o primeiro dia da semana é terça-feira, a semana 1 começa em 28 de dezembro de 2021 e termina na segunda-feira, 3 de janeiro de 2022.

Exemplo 5: Exemplo de objeto de gráfico

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém o mesmo conjunto de dados e cenário do primeiro exemplo.

No entanto, neste exemplo, o conjunto de dados inalterado é carregado no aplicativo. O cálculo que retorna o número da semana é criado como uma medida em um objeto de gráfico.

Script de carregamento

```
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
];
```

Resultados

Faça o seguinte:

1. Carregue os dados e abra uma pasta. Crie uma nova tabela.
2. Adicione os seguintes campos como dimensões:
    - `id`
    - `date`

3. Em seguida, crie a seguinte medida:
   `=week (date)`
4. Crie uma medida, `week_day` para mostrar o valor do dia da semana de cada data de transação:
   `=weekday(date)`

Tabela de resultados

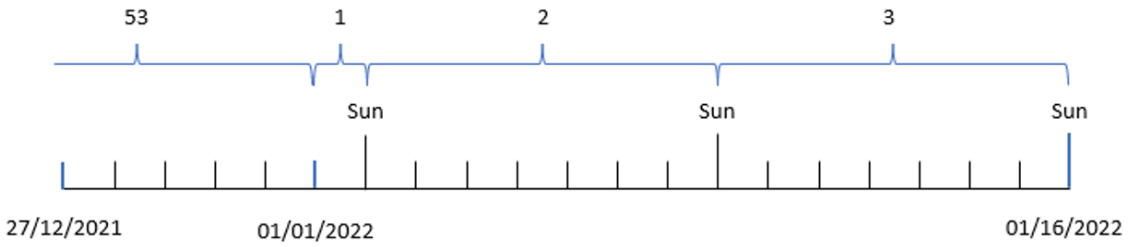
id	date	=week(date)	=weekday(date)
8183	12/27/2021	53	Seg
8184	12/28/2021	53	Ter
8185	12/29/2021	53	Qua
8186	12/30/2021	53	Qui
8187	12/31/2021	53	Sex
8188	01/01/2022	1	Sáb
8189	01/02/2022	2	Dom
8190	01/03/2022	2	Seg
8191	01/04/2022	2	Ter
8192	01/05/2022	2	Qua
8193	01/06/2022	2	Qui
8194	01/07/2022	2	Sex
8195	01/08/2022	2	Sáb
8196	01/09/2022	3	Dom
8197	01/10/2022	3	Seg
8198	01/11/2022	3	Ter
8199	01/12/2022	3	Qua
8200	01/13/2022	3	Qui
8201	01/14/2022	3	Sex

O campo `week_number` é criado na instrução de carregamento anterior usando a função `week()` e transmitindo o campo `date` como o argumento da função.

Nenhum outro parâmetro é informado para a função e, portanto, as seguintes variáveis padrão que afetam a função `week()` estão em vigor:

- `BrokenWeeks`: A contagem semanal começa em 1º de janeiro
- `FirstWeekDay`: O primeiro dia da semana é domingo

Diagrama da função week(), exemplo de objeto de gráfico



Como o aplicativo está usando a variável de sistema `BrokenWeeks` padrão, a semana 1 começa em 1º de janeiro, um sábado.

Por causa da variável de sistema `FirstWeekDay` padrão, as semanas começam em um domingo. O primeiro domingo após 1º de janeiro ocorre em 2 de janeiro, que é quando começa a semana 2.

Exemplo 6: cenário

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações para a última semana de 2019 e as duas primeiras semanas de 2020. Esse conjunto é carregado em uma tabela denominada `Transactions`.
- O campo de data fornecido no formato da variável de sistema `DateFormat` (MM/DD/AAAA).

O aplicativo usa principalmente semanas interrompidas em seu painel. No entanto, o usuário final deseja um objeto de gráfico que apresentasse o total de vendas por semana usando semanas ininterruptas. O dia de referência deve ser 2 de janeiro, com semanas começando em uma terça-feira. Isso pode ser alcançado mesmo quando essa dimensão não está disponível no modelo de dados. Para isso, use a função `week()` como uma dimensão calculada no gráfico.

Script de carregamento

```
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
];
```

Resultados

Faça o seguinte:

1. Carregue os dados e abra uma pasta. Crie uma nova tabela.
2. Crie a seguinte dimensão calculada:
   `=week(date)`
3. Em seguida, crie a seguinte medida de agregação:
   `=sum(amount)`
4. Defina o **Formato numérico** da medida como **Dinheiro**.
5. Selecione o menu **Classificação** e, para a dimensão calculada, remova a classificação personalizada.
6. Desmarque as opções **Classificar numericamente** e **Classificar alfabeticamente**.

Tabela de resultados

week(date)	sum(amount)
52	$125.69
53	$146.42
1	$200.09
2	$347.57
3	$122.01

weekday

Esta função retorna um valor dual com:

- Um nome do dia conforme definido na variável de ambiente **DayNames**.
- Um inteiro entre 0 e 6 correspondendo ao dia nominal da semana (0-6).

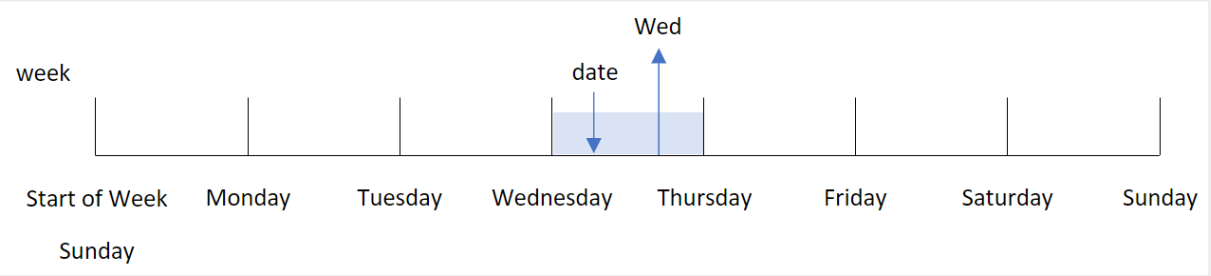
Sintaxe:

```
weekday(date [,first_week_day=0])
```

Tipo de dados de retorno: dual

A função `weekday()` determina em qual dia da semana uma data ocorre. Em seguida, ela retorna um valor de string representando esse dia.

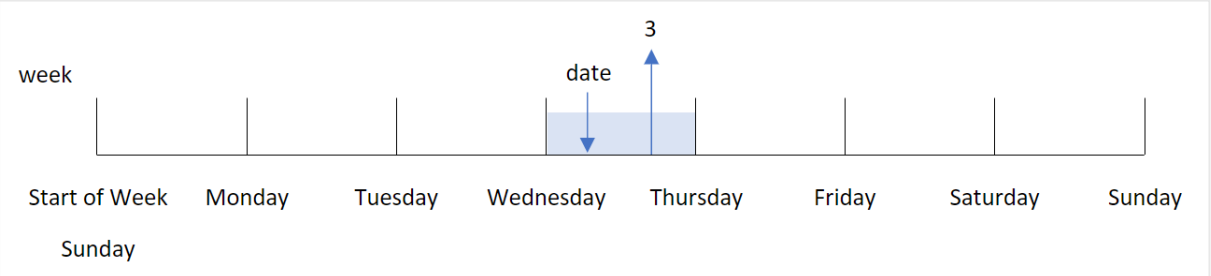
Diagrama da função weekday() que retorna o nome do dia em que uma data cai



O resultado retorna o valor numérico correspondente a esse dia da semana (0-6), com base no dia de início da semana. Por exemplo, se o primeiro dia da semana for definido como domingo, uma quarta-feira retornará um valor numérico de `3`. Esse dia de início é determinado pela `variável` de sistema FirstWeekDay ou pelo parâmetro da função `first_week_day`.

Você pode usar esse valor numérico como parte de uma expressão aritmética. Por exemplo, multiplique por `1` para retornar o valor em si.

Diagrama da função weekday() com o valor numérico do dia sendo mostrado no lugar do nome do dia



Quando usar

A função `weekday()` é útil quando você deseja comparar agregações por dia da semana. Por exemplo, se você quiser comparar a média de vendas de produtos por dia da semana.

Essas dimensões podem ser criadas no script de carregamento usando a função para criar um campo em uma tabela de **Calendário mestre** ou podem ser criadas diretamente em um gráfico como uma medida calculada.

Tópicos relacionados

Tópicos	Interação
FirstWeekDay (page 226)	Define o dia de início de cada semana.

Argumentos

Argumento	Descrição
date	A data ou o carimbo de data/hora a ser avaliado.
first_week_day	Especifica o dia no qual inicia a semana. Se omitido, o valor da variável **FirstWeekDay** é usado. *FirstWeekDay (page 226)*

Você pode usar os seguintes valores para definir o dia em que a semana começa no argumento `first_week_day`:

Valores de first_week_day

Dia	Valor
Segunda-feira	0
Terça-feira	1
Quarta-feira	2
Quinta-feira	3
Sexta-feira	4
Sábado	5
Domingo	6

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Salvo indicação em contrário, `FirstWeekDay` é definido como `0` nesses exemplos.

Exemplos de funções

Exemplo	Resultado
`weekday('10/12/1971')`	Retorna “True” e 1.
`weekday('10/12/1971' , 6)`	Retorna “True” e 2. Neste exemplo, domingo (6) é o primeiro dia da semana.

`SET FirstWeekDay=6;`
`...`
 	
`weekday('10/12/1971')`
	Retorna “True” e 2.

Exemplo 1: sequência de caracteres para dias da semana

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações para 2022, que é carregado em uma tabela denominada "`Transactions`".
- A variável de sistema `FirstWeekDay` que é definida como 6 (domingo).
- A variável `DayNames` que está configurada para usar os nomes de dias padrão.
- Um carregamento anterior que contém a função `weekday()`, que é definida como o campo "`week_day`" e retorna o dia da semana em que as transações ocorreram.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `id`
- `date`
- `week_day`

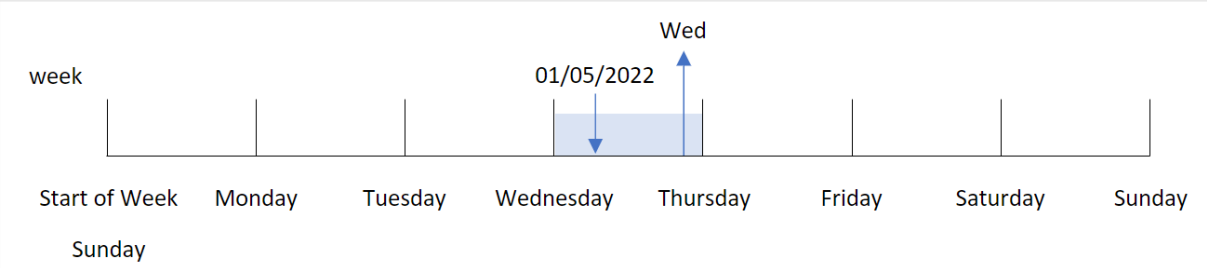
Tabela de resultados

id	date	week_day
8188	01/01/2022	Sáb
8189	01/02/2022	Dom
8190	01/03/2022	Seg
8191	01/04/2022	Ter
8192	01/05/2022	Qua
8193	01/06/2022	Qui
8194	01/07/2022	Sex

O campo "`week_day`" é criado na instrução de carregamento anterior usando a função `weekday()` e transmitindo o campo de data como o argumento da função.

A função `weekday()` retorna o valor da cadeia de caracteres do dia da semana; ou seja, ela retorna o nome do dia da semana que é definido pela variável de sistema `DayNames`.

Diagrama da função `weekday()` que retorna quarta-feira como o dia da semana para a transação 8192



A transação 8192 ocorreu em 5 de janeiro. A variável de sistema `FirstWeekDay` define o primeiro dia da semana como domingo. A transação da função `weekday()` ocorreu em uma quarta-feira e retorna esse valor, na forma abreviada da variável de sistema `DayNames`, no campo `week_day`.

Os valores no campo "week_day" estão alinhados à direita na coluna porque há um resultado duplo de número e texto para o campo (quarta-feira, 3). Para converter o valor do campo em seu equivalente numérico, o campo pode ser encapsulado dentro da função num(). Por exemplo, na Transação 8192, o valor de quarta-feira seria convertido no número 3.

Exemplo 2: first_week_day

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações para 2022, que é carregado em uma tabela chamada Transactions.
- A variável de sistema FirstWeekDay que é definida como 6 (domingo).
- A variável DayNames que está configurada para usar os nomes de dias padrão.
- Uma carga anterior que contém a função weekday(), que é definida como o campo "week_day" e retorna o dia da semana em que as transações ocorreram.

Script de carregamento

```
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
];
```

Resultados

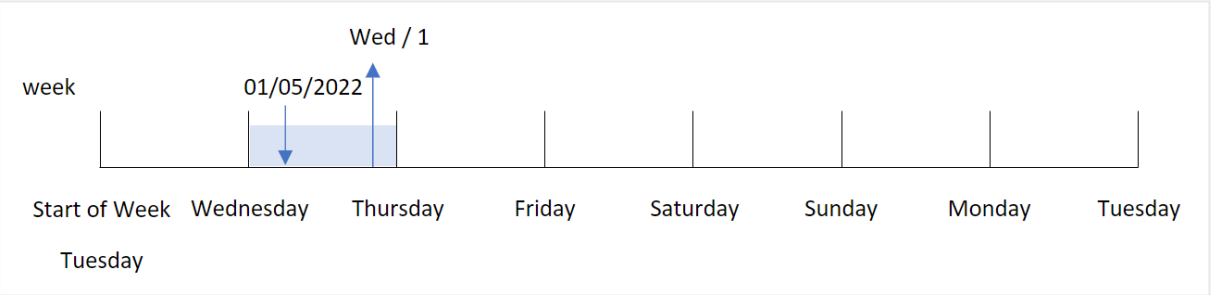
Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- id
- date
- week_day

Tabela de resultados

id	date	week_day
8188	01/01/2022	Sáb
8189	01/02/2022	Dom
8190	01/03/2022	Seg
8191	01/04/2022	Ter
8192	01/05/2022	Qua
8193	01/06/2022	Qui
8194	01/07/2022	Sex

Diagrama da função weekday() que mostra que quarta-feira tem o valor de número duplo de 1



Como o argumento first_week_day está definido como 1 na função weekday(), o primeiro dia da semana é terça-feira. Portanto, todas as transações que ocorrerem em uma terça-feira terão um valor numérico duplo de 0.

A transação 8192 ocorreu em 5 de janeiro. A função weekday() identifica que esta é uma quarta-feira e, portanto, a expressão retornaria o valor de número duplo de 1.

Exemplo 3: Exemplo de objeto de gráfico

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações para 2022, que é carregado em uma tabela denominada "Transactions".

- A variável de sistema `FirstWeekDay` que é definida como `6` (domingo).
- A variável `DayNames` que está configurada para usar os nomes de dias padrão.

No entanto, neste exemplo, o conjunto de dados permanece inalterado e é carregado no aplicativo. O cálculo que identifica o valor do dia da semana é criado como uma medida em um gráfico no aplicativo.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `id`
- `date`

Para calcular o valor do dia da semana, crie a seguinte medida:

- `=weekday(date)`

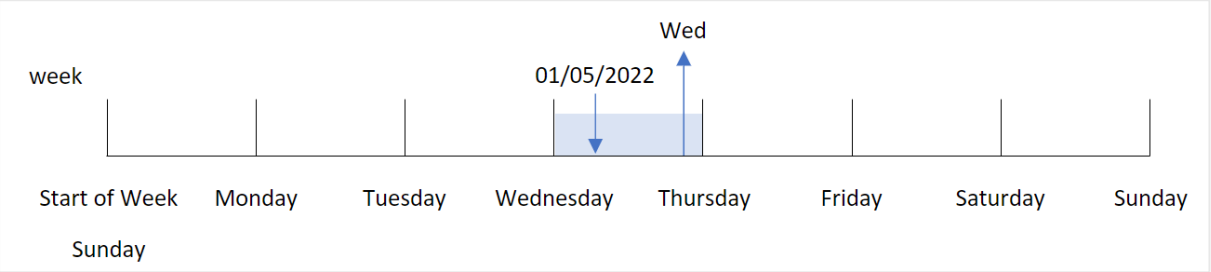
Tabela de resultados

id	date	=weekday(date)
8188	01/01/2022	Sáb
8189	01/02/2022	Dom
8190	01/03/2022	Seg
8191	01/04/2022	Ter
8192	01/05/2022	Qua
8193	01/06/2022	Qui
8194	01/07/2022	Sex

O campo "=weekday(date)" é criado no gráfico usando a função weekday() e transmitindo o campo de data como o argumento da função.

A função weekday() retorna o valor da cadeia de caracteres do dia da semana; ou seja, ela retorna o nome do dia da semana que é definido pela variável de sistema DayNames.

Diagrama da função weekday() que retorna quarta-feira como o dia da semana para a transação 8192



A transação 8192 ocorreu em 5 de janeiro. A variável de sistema FirstWeekDay define o primeiro dia da semana como domingo. A transação da função weekday() ocorreu em uma quarta-feira e retorna esse valor, na forma abreviada da variável de sistema DayNames, no campo =weekday(date).

Exemplo 4: Cenário

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações para 2022, que é carregado em uma tabela denominada "Transactions".
- A variável de sistema FirstWeekDay que é definida como 6 (domingo).
- A variável DayNames que está configurada para usar os nomes de dias padrão.

O usuário final gostaria de um gráfico que apresentasse a média de vendas por dia da semana para as transações.

Script de carregamento

```
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);
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `=weekday(date)`
- `=avg(amount)`

Defina o **Formato numérico** da medida como **Dinheiro**.

Tabela de resultados

weekday(date)	Avg(amount)
Dom	$536.96
Seg	$500.80
Ter	$515.63
Qua	$509.21
Qui	$482.70
Sex	$441.33
Sáb	$505.22

weekend

Essa função retorna um valor correspondente a um carimbo de data/hora do último milissegundo do último dia da semana do calendário contendo **date**. O formato de saída padrão será o **DateFormat** definido no script.

Sintaxe:

```
WeekEnd(timestamp [, period_no [, first_week_day ]])
```

Tipo de dados de retorno: dual

A função `weekend()` determina em qual semana a data cai. Em seguida, ela retorna um carimbo de data/hora, em formato de data, para o último milissegundo daquela semana. O primeiro dia da semana é determinado pela `variável` de ambiente FirstWeekDay. No entanto, isso pode ser substituído pelo argumento `first_week_day` na função `weekend()`.

Argumentos

Argumento	Descrição
timestamp	A data ou o carimbo de data/hora a ser avaliado.

Argumento	Descrição
period_no	**shift** é um inteiro, em que o valor 0 indica a semana que contém a **date**. Os valores negativos no deslocamento indicam semanas precedentes e os valores positivos indicam semanas subsequentes.
first_week_day	Especifica o dia no qual inicia a semana. Se omitido, o valor da variável **FirstWeekDay** é usado. Os valores possíveis para **first_week_day** são 0 para segunda-feira, 1 para terça, 2 para quarta-feira, 3 para quinta-feira, 4 para sexta-feira, 5 para sábado e 6 para domingo. Para obter mais informações sobre a variável de sistema, consulte *FirstWeekDay (page 226)*

Quando usar

A função `weekend()` é normalmente usada como parte de uma expressão quando o usuário deseja que o cálculo use os dias restantes da semana para a data especificada. Por exemplo, ela pode ser usada se um usuário deseja calcular o total de juros ainda não acumulados durante a semana.

Os exemplos a seguir pressupõem que:

```
SET FirstWeekDay=0;
```

Exemplo	Resultado
`weekend('01/10/2013')`	Retorna 01/12/2013 23:59:59.
`weekend('01/10/2013', -1)`	Retorna 01/05/2013 23:59:59..
`weekend('01/10/2013', 0, 1)`	Retorna 01/14/2013 23:59:59.

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplos:

Se quiser configurações ISO para semanas e números de semanas, certifique-se de ter o seguinte no script:

```
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
```

Se quiser configurações dos EUA, certifique-se de ter o seguinte no script:

```
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
```

Os exemplos acima resultam no seguinte resultado da função `weekend()`:

Exemplo de função de fim de semana

Date	ISO week end	US week end
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
Ter 2021, 5 de janeiro	2021-01-10	1/9/2021

Os fins de semana são aos domingos na coluna ISO e aos sábados na coluna US.

Exemplo 1: Exemplo básico

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações para 2022, que é carregado em uma tabela denominada `Transactions`.
- O campo de data fornecido no formato da variável de sistema `DateFormat`(MM/DD/AAAA).
- A criação de um campo, `end_of_week`, que retorna um carimbo de data/hora para o final da semana em que as transações ocorreram.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `end_of_week`
- `end_of_week_timestamp`

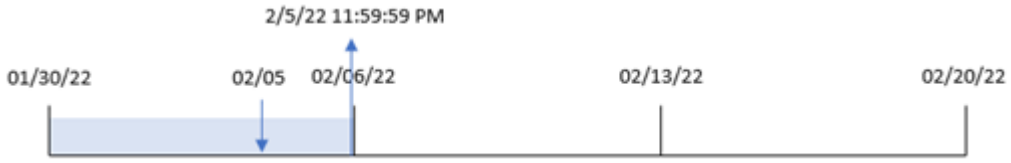
Tabela de resultados

date	end_of_week	end_of_week_timestamp
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
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

O campo `end_of_week` é criado na instrução de carregamento anterior usando a função `weekend()` e transmitindo o campo de data como o argumento da função.

A função `weekend()` identifica em qual semana o valor da data cai e retorna um carimbo de data/hora para o último milissegundo dessa semana.

*Diagrama da função **weekend()**, exemplo básico*



A transação 8191 ocorreu em 5 de fevereiro. A variável de sistema `FirstWeekDay` define o primeiro dia da semana como domingo. A função `weekend()` identifica que o primeiro sábado após 5 de fevereiro (e, portanto, o final da semana) foi em 5 de fevereiro. Portanto, o valor `end_of_week` dessa transação retorna o último milissegundo desse dia, que é 5 de fevereiro às 11:59:59 PM.

Exemplo 2: period_no

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- O mesmo conjunto de dados e cenário do primeiro exemplo.
- A criação de um campo `previous_week_end`, que retorna o carimbo de data/hora do início da semana antes da transação.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

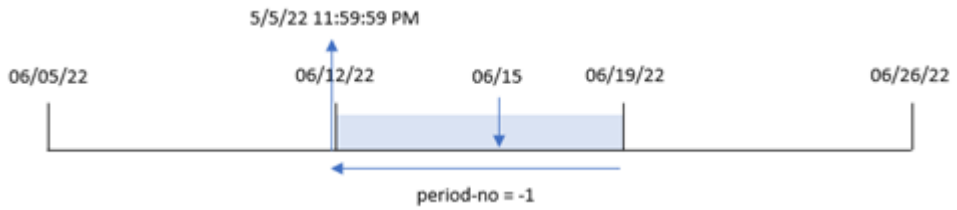
- `date`
- `previous_week_end`
- `previous_week_end_timestamp`

Tabela de resultados

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

Nesse caso, como um `period_no` de -1 foi usado como argumento de compensação na função `weekend()`, a função primeiro identifica a semana em que as transações ocorrem. Em seguida, ela procura uma semana antes e identifica o último milissegundo daquela semana.

Diagrama da função weekend(), exemplo de period_no



A transação 8196 ocorreu em 15 de junho. A função `weekend()` identifica que a semana começa em 12 de junho. Portanto, a semana anterior termina em 11 de junho às 11:59:59 PM; esse é o valor retornado para o campo `previous_week_end`.

Exemplo 3: first_week_day

Script de carregamento e resultados

Visão geral

Abra o editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém o mesmo conjunto de dados e cenário do primeiro exemplo. No entanto, neste exemplo, precisamos definir terça-feira como o primeiro dia da semana de trabalho.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

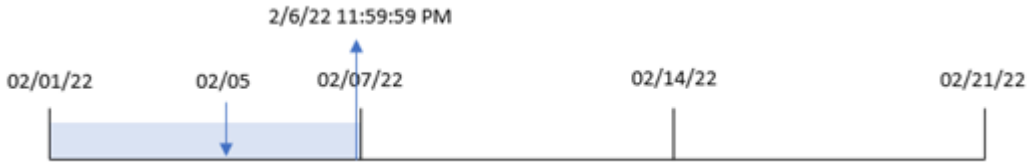
- `date`
- `end_of_week`
- `end_of_week_timestamp`

Tabela de resultados

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
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

Nesse caso, como o argumento `first_week_date` de 1 é usado na função `weekend()`, ele define o primeiro dia da semana como terça-feira.

Diagrama da função weekend(), exemplo de first_week_day



A transação 8191 ocorreu em 5 de fevereiro. A função `weekend()` identifica que a primeira segunda-feira após essa data (e, portanto, o final da semana e o valor retornado) foi em 6 de fevereiro às 11:59:59 PM.

Exemplo 4: Exemplo de objeto de gráfico

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém o mesmo conjunto de dados e cenário do primeiro exemplo. No entanto, neste exemplo, o conjunto de dados inalterado é carregado no aplicativo. O cálculo que retorna um carimbo de data/hora para o final da semana em que as transações ocorreram é criado como uma medida em um objeto de gráfico do aplicativo.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão: `date`.

Para calcular o início da semana em que uma transação ocorre, adicione as seguintes medidas:

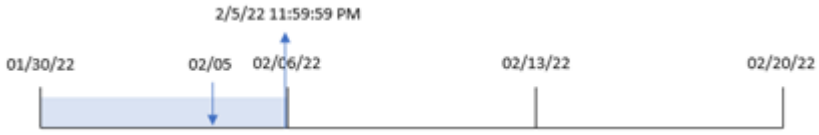
- `=weekend(date)`
- `=timestamp(weekend(date))`

Tabela de resultados

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
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

A medida `end_of_week` é criada no objeto de gráfico usando a função `weekend()` e transmitindo o campo de data como o argumento da função. A função `weekend()` identifica em qual semana o valor da data cai, retornando um carimbo de data/hora para o último milissegundo dessa semana.

Diagrama da função weekend(), exemplo de objeto de gráfico



A transação 8191 ocorreu em 5 de fevereiro. A variável de sistema `FirstWeekDay` define o primeiro dia da semana como domingo. A função `weekend()` identifica que o primeiro sábado após 5 de fevereiro (e, portanto, o final da semana) foi em 5 de fevereiro. Portanto, o valor `end_of_week` dessa transação retorna o último milissegundo desse dia, que é 5 de fevereiro às 11:59:59 PM.

Exemplo 5: Cenário

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados que é carregado em uma tabela denominada `Employee_Expenses`.
- Dados que consistem em IDs de funcionários, nomes de funcionários e a média diária de reivindicações de despesas de cada funcionário.

O usuário final deseja um objeto de gráfico que mostre, por ID de funcionário e nome de funcionário, as reivindicações de despesas estimadas ainda a serem acumuladas para o restante da semana.

Script de carregamento

```
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
];
```

Resultados

Faça o seguinte:

1. Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:
   - `employee_id`
   - `employee_name`
2. Em seguida, crie uma medida para calcular os juros acumulados:
   `=(weekend(today(1))-today(1))*avg_daily_claim`
3. Defina o **Formato numérico** da medida como **Dinheiro**.

Tabela de resultados

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

A função `weekend()`, usando a data de hoje como seu único argumento, retorna a data de término da semana atual. Em seguida, subtraindo a data de hoje da data de término da semana, a expressão retorna o número de dias que restam nessa semana.

Esse valor é então multiplicado pela média de solicitações de despesas diárias por cada funcionário para calcular o valor estimado das solicitações que cada funcionário deve fazer na semana restante.

weekname

Esta função retorna um valor que mostra o número do ano e da semana com um valor numérico subjacente que corresponde a um carimbo de hora do primeiro milissegundo do primeiro dia da semana que contém a **date**.

Sintaxe:

```
WeekName(date[, period_no [, first_week_day [, broken_weeks [, reference_
day]]]])
```

A função `weekname()` determina em qual semana a data cai e retorna o número e o ano dessa semana. O primeiro dia da semana é determinado pela `variável` de sistema FirstWeekDay. No entanto, você também pode alterar o primeiro dia da semana usando o argumento `first_week_day` na função `weekname()`.

No Qlik Sense, as configurações regionais são obtidas quando o aplicativo é criado, e as configurações correspondentes são armazenadas no script como variáveis de ambiente.

Um desenvolvedor de aplicativos norte-americano geralmente obtém `Set BrokenWeeks=1;` no script, o que corresponde a semanas interrompidas. Um desenvolvedor de aplicativos europeu geralmente obtém `Set BrokenWeeks=0;` no script, correspondendo a semanas ininterruptas.

Se seu aplicativo usar semanas interrompidas, a contagem de números de semana começará em 1º de janeiro e terminará no dia anterior à variável do sistema `FirstWeekDay`, independentemente de quantos dias tenham ocorrido.

No entanto, se o seu aplicativo estiver usando semanas não quebradas, a semana 1 pode começar no ano anterior ou nos primeiros dias de janeiro. Isso depende de como você usa as variáveis de sistema `ReferenceDay` e `FirstWeekDay`.

Exemplo da função Weekname

Date	**Nome da semana ISO**	**Nome da semana dos EUA**
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
Ter 2021, 5 de janeiro	2021/01	2021/02

Quando usar

A função `weekname()` é útil para quando você deseja comparar agregações por semanas.

Por exemplo, se você quiser ver o total de vendas de produtos por semana. Para manter a consistência com a variável de ambiente `BrokenWeeks` no aplicativo, use `weekname()` em vez de `lunarweekname()`. Se o aplicativo estiver usando semanas não quebradas, a semana 1 pode conter datas de dezembro do ano anterior ou pode excluir datas em janeiro do ano atual. Se o aplicativo estiver usando semanas quebradas, a semana 1 pode conter menos de sete dias.

Tipo de dados de retorno: dual

Argumentos

Argumento	Descrição
timestamp	A data ou o carimbo de data/hora a ser avaliado.
period_no	**shift** é um inteiro, em que o valor 0 indica a semana que contém a **date**. Os valores negativos no deslocamento indicam semanas precedentes e os valores positivos indicam semanas subsequentes.
first_week_day	Especifica o dia no qual inicia a semana. Se omitido, o valor da variável **FirstWeekDay** é usado. Os valores possíveis **first_week_day** são 0 para segunda-feira, 1 para terça, 2 para quarta-feira, 3 para quinta-feira, 4 para sexta-feira, 5 para sábado e 6 para domingo. Para obter mais informações sobre a variável de sistema, consulte *FirstWeekDay (page 226)*.
broken_weeks	Se você não especificar **broken_weeks**, o valor da variável **BrokenWeeks** será usado para definir se as semanas são quebradas ou não.
reference_day	Se você não especificar **reference_day**, o valor da variável **ReferenceDay** será usado para definir qual dia em janeiro será definido como dia de referência para definir a semana 1. Por padrão, as funções Qlik Sense usam 4 como o dia de referência. Isso significa que a semana 1 deve conter 4 de janeiro, ou dito de outra forma, que a semana 1 deve sempre ter pelo menos 4 dias em janeiro.

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Os exemplos abaixo pressupõem que:

```
Set FirstWeekDay=0;
Set BrokenWeeks=0;
Set ReferenceDay=4;
```

Exemplos de funções

Exemplo	Resultado
weekname('01/12/2013')	Retorna 2013/02.
weekname('01/12/2013', -1)	Retorna 2013/01.
weekname('01/12/2013', 0, 1)	Retorna 2013/02.

Exemplo 1: data sem argumentos adicionais

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações para a última semana de 2021 e as duas primeiras semanas de 2022 é carregado em uma tabela chamada "`Transactions`".
- A variável de sistema `DateFormat` que é definida como o formato `MM/DD/YYYY`.
- A variável de sistema `BrokenWeeks` que é definida como `1`.
- A variável de sistema `FirstWeekDay` que é definida como `6`.
- Um carregamento anterior que contém o seguinte:
    - A função `weekday()` definida como o campo "`week_number`", que retorna o número do ano e da semana quando as transações ocorreram.
    - A função `weekname()` definida como o campo chamado "`week_day`" para mostrar o valor do dia da semana de cada data de transação.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `id`
- `date`
- `week_day`
- `week_number`

Tabela de resultados

id	date	week_day	week_number
8183	12/27/2021	Seg	2021/53
8184	12/28/2021	Ter	2021/53
8185	12/29/2021	Qua	2021/53
8186	12/30/2021	Qui	2021/53
8187	12/31/2021	Sex	2021/53
8188	01/01/2022	Sáb	2022/01
8189	01/02/2022	Dom	2022/02
8190	01/03/2022	Seg	2022/02
8191	01/04/2022	Ter	2022/02
8192	01/05/2022	Qua	2022/02
8193	01/06/2022	Qui	2022/02
8194	01/07/2022	Sex	2022/02
8195	01/08/2022	Sáb	2022/02

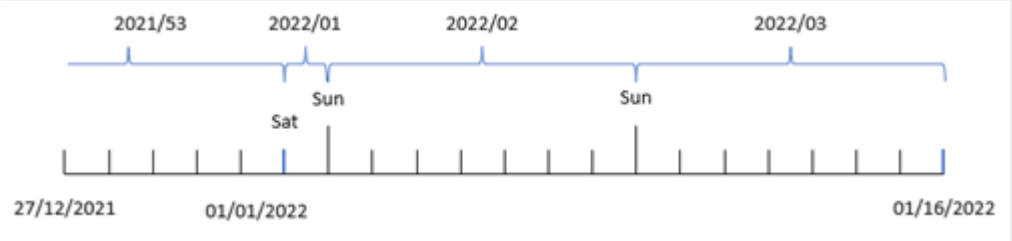
id	date	week_day	week_number
8196	01/09/2022	Dom	2022/03
8197	01/10/2022	Seg	2022/03
8198	01/11/2022	Ter	2022/03
8199	01/12/2022	Qua	2022/03
8200	01/13/2022	Qui	2022/03
8201	01/14/2022	Sex	2022/03

O campo "week_number" é criado na instrução de carregamento anterior usando a função weekname() e transmitindo o campo de data como o argumento da função.

A função weekname() identifica inicialmente em qual semana o valor da data cai e retorna a contagem do número da semana e o ano em que ocorre a transação.

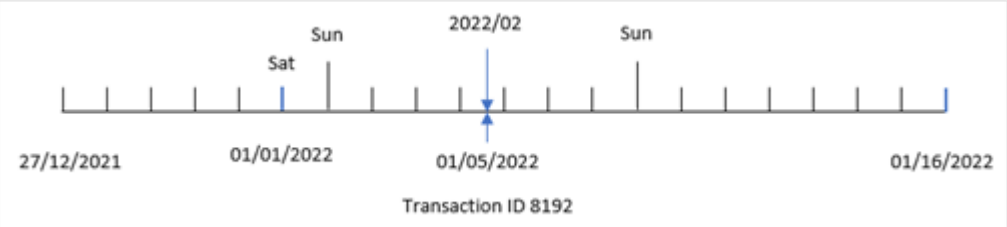
A variável de sistema FirstWeekDay define o domingo como o primeiro dia da semana. A variável de sistema BrokenWeeks define o aplicativo para usar semanas quebradas, o que significa que a semana 1 começará em 1º de janeiro.

Diagrama da função weekname() com as variáveis padrão.



A semana 1 começa em 1º de janeiro, que é um sábado e, portanto, as transações que ocorrem nessa data retornam o valor 2022/01 (o ano e o número da semana).

Diagrama da função weekname() que identifica o número da semana da transação 8192.



Como o aplicativo está usando semanas quebradas, e o primeiro dia da semana é domingo, as transações que ocorrem de 2 a 8 de janeiro retornam o valor 2022/02 (semana número 2 em 2022). Um exemplo disso seria a transação 8192, que ocorreu em 5 de janeiro e retorna o valor 2022/02 para o campo "week_number".

Exemplo 2: period_no

Script de carregamento e resultados

Visão geral

São usados o mesmo conjunto de dados e cenário do primeiro exemplo.

No entanto, neste exemplo, a tarefa é criar um campo, "previous_week_number", que retorna o ano e o número da semana, antes da ocorrência das transações.

Abra o Editor de carregamento de dados e adicione o script de carregamento a seguir a uma nova guia.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

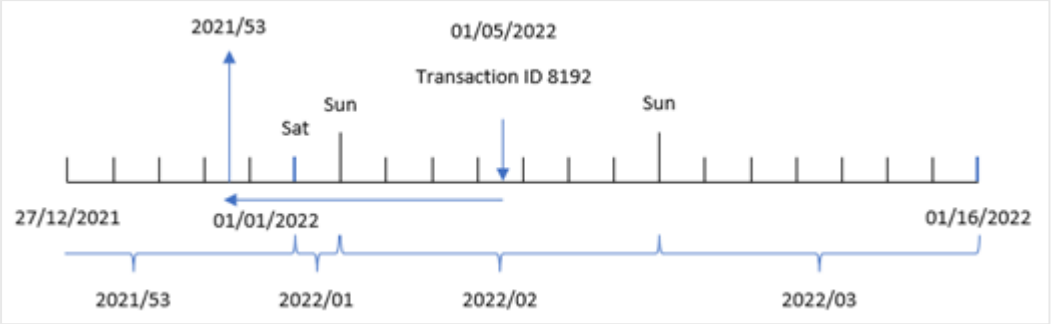
- id
- date
- week_day
- week_number

Tabela de resultados

id	date	week_day	week_number
8183	12/27/2021	Seg	2021/52
8184	12/28/2021	Ter	2021/52
8185	12/29/2021	Qua	2021/52
8186	12/30/2021	Qui	2021/52
8187	12/31/2021	Sex	2021/52
8188	01/01/2022	Sáb	2021/52
8189	01/02/2022	Dom	2021/53
8190	01/03/2022	Seg	2021/53
8191	01/04/2022	Ter	2021/53
8192	01/05/2022	Qua	2021/53
8193	01/06/2022	Qui	2021/53
8194	01/07/2022	Sex	2021/53
8195	01/08/2022	Sáb	2022/01
8196	01/09/2022	Dom	2022/02
8197	01/10/2022	Seg	2022/02
8198	01/11/2022	Ter	2022/02
8199	01/12/2022	Qua	2022/02
8200	01/13/2022	Qui	2022/02
8201	01/14/2022	Sex	2022/02

Como um `period_no` de `-1` foi usado como o argumento de deslocamento na função `weekname()`, a função primeiro identifica a semana em que as transações ocorrem. Em seguida, ela procura uma semana antes e identifica o primeiro milissegundo daquela semana.

Diagrama da função weekname() com um deslocamento de period_no de -1.



A transação 8192 ocorreu em 5 de janeiro de 2022. A função `weekname()` procura uma semana antes, 30 de dezembro de 2021, e retorna o número da semana e o ano dessa data: `2021/53`.

Exemplo 3: first_week_day

Script de carregamento e resultados

Visão geral

São usados o mesmo conjunto de dados e cenário do primeiro exemplo.

No entanto, neste exemplo, a política da empresa é que a semana de trabalho comece na terça-feira.

Abra o Editor de carregamento de dados e adicione o script de carregamento a seguir a uma nova guia.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

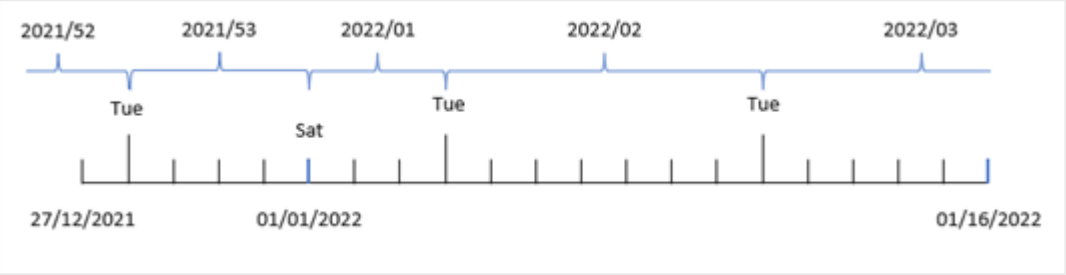
- `id`
- `date`
- `week_day`
- `week_number`

Tabela de resultados

id	date	week_day	week_number
8183	12/27/2021	Seg	2021/52
8184	12/28/2021	Ter	2021/53
8185	12/29/2021	Qua	2021/53
8186	12/30/2021	Qui	2021/53
8187	12/31/2021	Sex	2021/53
8188	01/01/2022	Sáb	2022/01
8189	01/02/2022	Dom	2022/01
8190	01/03/2022	Seg	2022/01
8191	01/04/2022	Ter	2022/02
8192	01/05/2022	Qua	2022/02
8193	01/06/2022	Qui	2022/02
8194	01/07/2022	Sex	2022/02
8195	01/08/2022	Sáb	2022/02
8196	01/09/2022	Dom	2022/02
8197	01/10/2022	Seg	2022/02
8198	01/11/2022	Ter	2022/03
8199	01/12/2022	Qua	2022/03

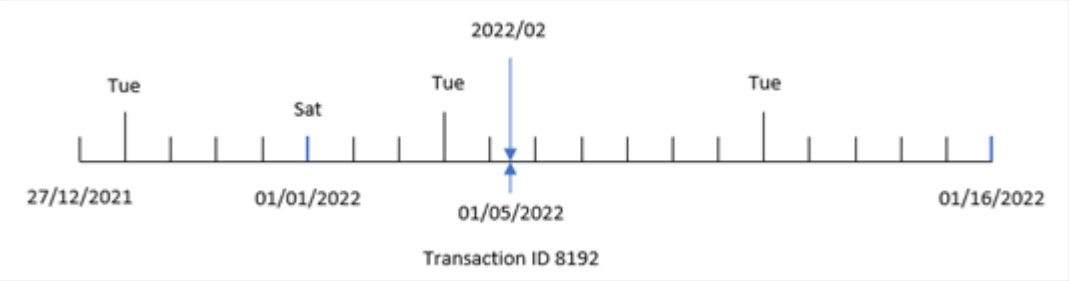
id	date	week_day	week_number
8200	01/13/2022	Qui	2022/03
8201	01/14/2022	Sex	2022/03

Diagrama da função weekname() com a terça-feira como o primeiro dia da semana.



Como o argumento `first_week_date` de `1` é usado na função `weekname()`, ele usa a terça-feira como o primeiro dia da semana. Portanto, a função determina que a semana 53 de 2021 começa na terça-feira, 28 de dezembro e, devido ao uso de semanas quebradas pelo aplicativo, a semana 1 começa em 1º de janeiro de 2022 e termina no último milissegundo de segunda-feira, 3 de janeiro de 2022.

Diagrama mostrando o número da semana da transação 8192 com a terça-feira como o primeiro dia da semana.



A transação 8192 ocorreu em 5 de janeiro de 2022. Portanto, usando um parâmetro `first_week_day` de terça-feira, a função `weekname()` retorna o valor `2022/02` para o campo "`week_number`".

Exemplo 4: Exemplo de objeto de gráfico

Script de carregamento e expressão de gráfico

Visão geral

São usados o mesmo conjunto de dados e cenário do primeiro exemplo.

No entanto, neste exemplo, o conjunto de dados permanece inalterado e é carregado no aplicativo. O cálculo que retorna o número do ano da semana para quando as transações ocorreram é criado como uma medida em um objeto de gráfico do aplicativo.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `id`
- `date`
- `=week_day (date)`

Para calcular o início da semana em que uma transação ocorre, crie a seguinte medida:

```
=weekname(date)
```

Tabela de resultados

id	date	=weekday(date)	=weekname(date)
8183	12/27/2021	Seg	2021/53
8184	12/28/2021	Ter	2021/53
8185	12/29/2021	Qua	2021/53
8186	12/30/2021	Qui	2021/53

id	date	=weekday(date)	=weekname(date)
8187	12/31/2021	Sex	2021/53
8188	01/01/2022	Sáb	2022/01
8189	01/02/2022	Dom	2022/02
8190	01/03/2022	Seg	2022/02
8191	01/04/2022	Ter	2022/02
8192	01/05/2022	Qua	2022/02
8193	01/06/2022	Qui	2022/02
8194	01/07/2022	Sex	2022/02
8195	01/08/2022	Sáb	2022/02
8196	01/09/2022	Dom	2022/03
8197	01/10/2022	Seg	2022/03
8198	01/11/2022	Ter	2022/03
8199	01/12/2022	Qua	2022/03
8200	01/13/2022	Qui	2022/03
8201	01/14/2022	Sex	2022/03

O campo "week_number" é criado como uma medida no objeto de gráfico usando a função `weekname()` e transmitindo o campo de data como argumento dessa função.

A função `weekname()` identifica inicialmente em qual semana o valor da data cai e retorna a contagem do número da semana e o ano em que ocorre a transação.

A variável de sistema `FirstWeekDay` define o domingo como o primeiro dia da semana. A variável de sistema `BrokenWeeks` define o aplicativo para usar semanas quebradas, o que significa que a semana 1 começa em 1º de janeiro.

Diagrama mostrando o número da semana com domingo como o primeiro dia da semana.

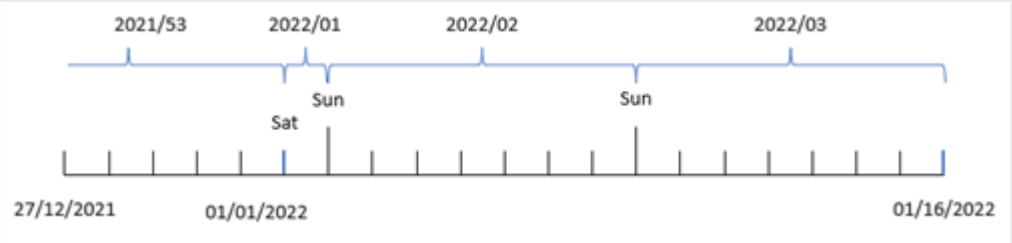
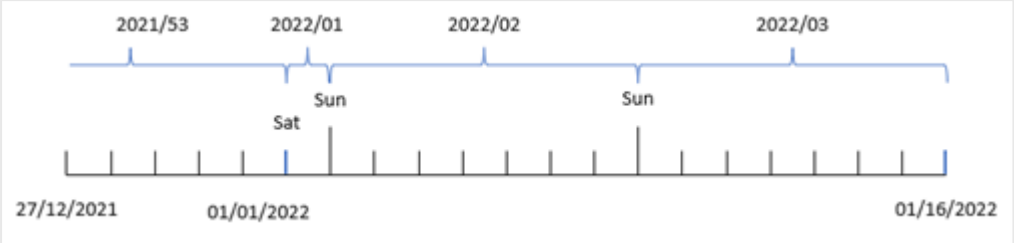


Diagrama mostrando que a transação 8192 ocorreu na semana número dois.



Como o aplicativo está usando semanas quebradas, e o primeiro dia da semana é domingo, as transações que ocorrerem de 2 a 8 de janeiro retornarão o valor 2022/02, a semana número 2 em 2022. Observe que a transação 8192 ocorreu em 5 de janeiro e retorna o valor 2022/02 para o campo "week_number".

Exemplo 5: Cenário

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações para a última semana de 2019 e as duas primeiras semanas de 2020 é carregado em uma tabela chamada "Transactions".
- A variável de sistema BrokenWeeks que é definida como 0.
- A variável de sistema ReferenceDay que é definida como 2.
- A variável de sistema DateFormat que é definida como o formato MM/DD/YYYY.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela.

Crie uma dimensão calculada usando a seguinte expressão:

```
=weekname(date)
```

Para calcular o total de vendas, crie a seguinte medida de agregação:

```
=sum(amount)
```

Defina o **Formato numérico** da medida como **Dinheiro**.

Tabela de resultados

weekname(date)	=sum(amount)
2019/52	$125.69
2020/01	$346.51
2020/02	$347.57
2020/03	$122.01

Para demonstrar os resultados do uso da função weekname() nesse cenário, adicione o seguinte campo como uma dimensão:

```
date
```

Tabela de resultados com o campo de data

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

Como o aplicativo usa semanas não quebradas, e a semana 1 exige no mínimo dois dias em janeiro devido à variável de sistema `ReferenceDay`, a semana 1 de 2020 inclui transações de 29 de dezembro de 2019.

weekstart

Esta função retorna um valor correspondente ao carimbo de data/hora com o primeiro milissegundo do primeiro dia da semana do calendário que contém a **date**. O formato de saída padrão é o **DateFormat** definido no script.

Sintaxe:

```
WeekStart(timestamp [, period_no [, first_week_day ]])
```

Tipo de dados de retorno: dual

A função `weekstart()` determina em qual semana a data cai. Em seguida, ela retorna um carimbo de data/hora, em formato de data, para o primeiro milissegundo daquela semana. O primeiro dia da semana é determinado pela `variável` de ambiente FirstWeekDay. No entanto, isso pode ser substituído pelo argumento `first_week_day` na função `weekstart()`.

Argumentos

Argumento	Descrição
timestamp	A data ou o carimbo de data/hora a ser avaliado.

Argumento	Descrição
period_no	**shift** é um inteiro, em que o valor 0 indica a semana que contém a **date**. Os valores negativos no deslocamento indicam semanas precedentes e os valores positivos indicam semanas subsequentes.
first_week_day	Especifica o dia no qual inicia a semana. Se omitido, o valor da variável **FirstWeekDay** é usado. Os valores possíveis **first_week_day** são 0 para segunda-feira, 1 para terça, 2 para quarta-feira, 3 para quinta-feira, 4 para sexta-feira, 5 para sábado e 6 para domingo. Para obter mais informações sobre a variável de sistema, consulte *FirstWeekDay (page 226)*.

Quando usar

A função `weekstart()` é comumente usada como parte de uma expressão quando o usuário deseja que o cálculo use a fração da semana decorrida até o momento. Por exemplo, ela pode ser usada se um usuário deseja calcular o total de salários ganhos pelos funcionários na semana até o momento.

Os exemplos a seguir pressupõem que:

```
SET FirstWeekDay=0;
```

Exemplos de funções

Exemplo	Resultado
`weekstart('01/12/2013')`	Retorna 01/07/2013.
`weekstart('01/12/2013', -1 )`	Retorna 11/31/2012.
`weekstart('01/12/2013', 0, 1)`	Retorna 01/08/2013.

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplos:

Se quiser configurações ISO para semanas e números de semanas, certifique-se de ter o seguinte no script:

```
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
```

Se quiser configurações dos EUA, certifique-se de ter o seguinte no script:

```
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
```

Os exemplos acima resultam no seguinte resultado da função `weekstart()`:

Exemplo da função Weekstart

Date	**Início da semana ISO**	**Início da semana dos EUA**
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
Ter 2021, 5 de janeiro	2021-01-04	1/3/2021

O início da semana é às segundas-feiras na coluna ISO e aos domingos na coluna US.

Exemplo 1: Sem argumentos adicionais

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações para 2022, que é carregado em uma tabela denominada `Transactions`.
- O campo de data fornecido no formato da variável de sistema `DateFormat`(MM/DD/AAAA).

- A criação de um campo, `start_of_week`, que retorna um carimbo de data/hora para o início da semana em que as transações ocorreram.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `start_of_week`
- `start_of_week_timestamp`

Tabela de resultados

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
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

O campo `start_of_week` é criado na instrução de carregamento anterior usando a função `weekstart()` e transmitindo o campo de data como o argumento da função.

A função `weekstart()` identifica inicialmente em qual semana o valor da data se enquadra, retornando um carimbo de data/hora para o primeiro milissegundo dessa semana.

Diagrama da função `weekstart()`, exemplo sem argumentos adicionais

A transação 8191 ocorreu em 5 de fevereiro. A variável de sistema `FirstWeekDay` define o primeiro dia da semana como domingo. A função `weekstart()` identifica que o primeiro domingo antes de 5 de fevereiro (e, portanto, o início da semana) foi em 30 de janeiro. Portanto, o valor `start_of_week` dessa transação retorna o primeiro milissegundo desse dia, que é 30 de janeiro às 12:00:00.

Exemplo 2: period_no

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- O mesmo conjunto de dados e cenário do primeiro exemplo.
- A criação de um campo `previous_week_start`, que retorna o carimbo de data/hora do início do trimestre antes da transação.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

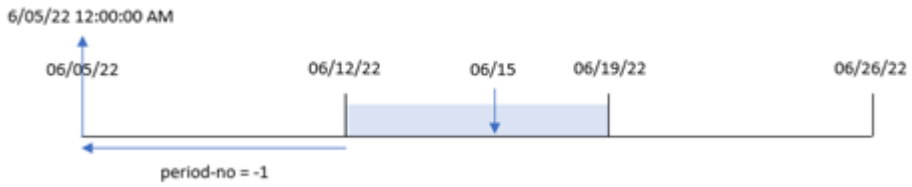
- `date`
- `previous_week_start`
- `previous_week_start_timestamp`

Tabela de resultados

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

Nesse caso, como um `period_no` de -1 foi usado como o argumento de deslocamento na função `weekstart()`, a função primeiro identifica a semana em que as transações ocorrem. Em seguida, ela procura uma semana antes e identifica o primeiro milissegundo daquela semana.

Diagrama da função weekstart(), exemplo de period_no



A transação 8196 ocorreu em 15 de junho. A função `weekstart()` identifica que a semana começa em 12 de junho. Portanto, a semana anterior começou em 5 de junho às 12h; esse é o valor retornado para o campo `previous_week_start`.

Exemplo 3: first_week_day

Script de carregamento e resultados

Visão geral

Abra o editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém o mesmo conjunto de dados e cenário do primeiro exemplo. No entanto, neste exemplo, precisamos definir terça-feira como o primeiro dia da semana de trabalho.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `start_of_week`
- `start_of_week_timestamp`

Tabela de resultados

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

Nesse caso, como o argumento `first_week_date` de 1 é usado na função `weekstart()`, ele define o primeiro dia da semana como terça-feira.

Diagrama da função `weekstart()`, exemplo de first_week_day

A transação 8191 ocorreu em 5 de fevereiro. A função `weekstart()` identifica que a primeira terça-feira antes dessa data (e, portanto, o início da semana e o valor retornado) foi 1º de fevereiro às 12h.

Exemplo 4: Exemplo de objeto de gráfico

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém o mesmo conjunto de dados e cenário do primeiro exemplo.

No entanto, neste exemplo, o conjunto de dados inalterado é carregado no aplicativo. O cálculo que retorna um carimbo de data/hora para o início da semana em que as transações ocorreram é criado como uma medida em um objeto de gráfico do aplicativo.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão: `date`.

Para calcular o início da semana em que uma transação ocorre, adicione as seguintes medidas:

- `=weekstart(date)`
- `=timestamp(weekstart(date))`

Tabela de resultados

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
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

date	start_of_week	start_of_week_timestamp
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

A medida `start_of_week` é criada no objeto de gráfico usando a função `weekstart()` e transmitindo o campo `date` como o argumento da função.

A função `weekstart()` identifica inicialmente em qual semana o valor da data se enquadra, retornando um carimbo de data/hora para o primeiro milissegundo dessa semana.

Diagrama da função `weekstart()`, exemplo de objeto de gráfico

A transação 8191 ocorreu em 5 de fevereiro. A variável de sistema `FirstWeekDay` define o primeiro dia da semana como domingo. A função `weekstart()` identifica que o primeiro domingo antes de 5 de fevereiro (e, portanto, o início da semana) foi 30 de janeiro. Portanto, o valor `start_of_week` dessa transação retorna o primeiro milissegundo desse dia, que é 30 de janeiro às 12h00.

Exemplo 5: Cenário

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados que é carregado em uma tabela denominada `Payroll`.
- Dados que consistem em IDs de funcionários, nomes de funcionários e o salário diário ganho por cada funcionário.

Os funcionários começam a trabalhar na segunda-feira e trabalham seis dias por semana. A variável de sistema `FirstWeekDay` não deve ser modificada.

O usuário final gostaria de um objeto de gráfico que exiba, por ID do funcionário e nome do funcionário, os salários ganhos na semana atual.

Script de carregamento

```
Payroll:
Load
*
Inline
[
employee_id,employee_name,day_rate
182,Mark, $150
183,Deryck, $125
184,Dexter, $125
185,Sydney,$270
186,Agatha,$128
];
```

Resultados

Faça o seguinte:

1. Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:
   - `employee_id`
   - `employee_name`
2. Em seguida, crie uma medida para calcular os salários ganhos na semana até o momento: `=if(today(1)-weekstart(today(1),0,0)<7,(today(1)-weekstart(today(1),0,0))*day_rate,day_rate*6)`
3. Defina o **Formato numérico** da medida como **Dinheiro**.

Tabela de resultados

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

A função `weekstart()`, ao usar a data de hoje como primeiro argumento e 0 como terceiro argumento, define segunda-feira como o primeiro dia da semana e retorna a data de início da semana atual. Ao subtrair esse resultado da data atual, a expressão retorna o número de dias decorridos até o momento nesta semana.

A condição então avalia se houve mais de seis dias nesta semana. Nesse caso, o `day_rate` do funcionário é multiplicado por 6 dias. Caso contrário, o `day_rate` é multiplicado pelo número de dias que ocorreram até o momento nesta semana.

weekyear

Essa função retorna o ano ao qual o número da semana pertence, de acordo com as variáveis de ambiente. O número da semana varia entre 1 e cerca de 52.

Sintaxe:

```
weekyear(timestamp [, first_week_day [, broken_weeks [, reference_day]]])
```

Tipo de dados de retorno: inteiro

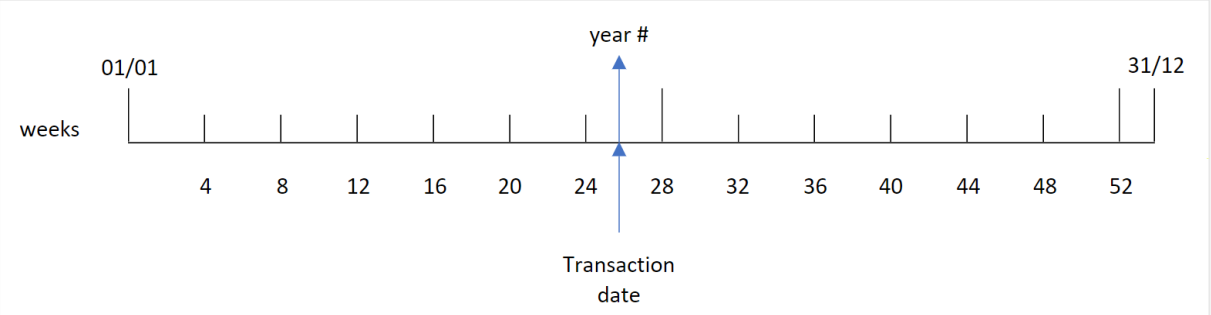
Argumentos

Argumento	Descrição
timestamp	A data ou o carimbo de data/hora a ser avaliado.
first_week_day	Especifica o dia no qual inicia a semana. Se omitido, o valor da variável **FirstWeekDay** será usado. Os valores possíveis **first_week_day** são 0 para segunda-feira, 1 para terça, 2 para quarta-feira, 3 para quinta-feira, 4 para sexta-feira, 5 para sábado e 6 para domingo. Para obter mais informações sobre a variável de sistema, consulte *FirstWeekDay (page 226)*.
broken_weeks	Se você não especificar **broken_weeks**, o valor da variável **BrokenWeeks** será usado para definir se as semanas são quebradas ou não.
reference_day	Se você não especificar **reference_day**, o valor da variável **ReferenceDay** será usado para definir qual dia em janeiro será definido como dia de referência para definir a semana 1. Por padrão, as funções Qlik Sense usam 4 como o dia de referência. Isso significa que a semana 1 deve conter 4 de janeiro, ou dito de outra forma, que a semana 1 deve sempre ter pelo menos 4 dias em janeiro.

A função `weekyear()` determina em qual semana do ano uma data se enquadra. Em seguida, ele retorna o ano correspondente a esse número da semana.

Se `BrokenWeeks` estiver definido como 0 (false), `weekyear()` retornará o mesmo que `year()`.

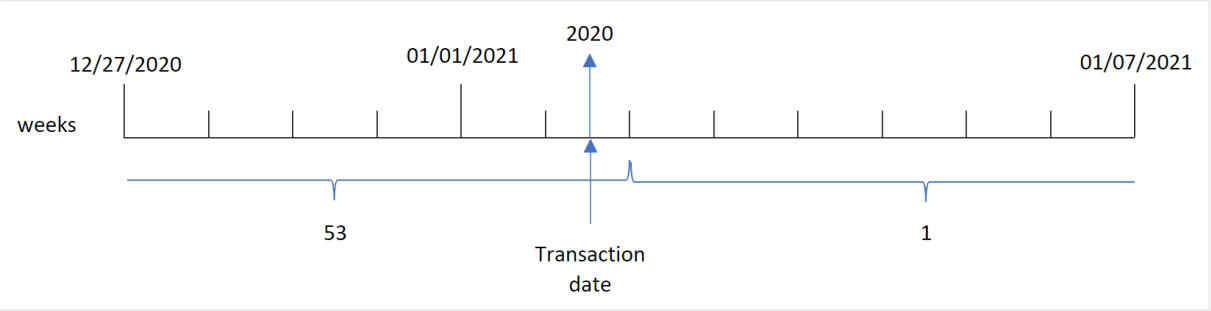
Diagrama do intervalo da função `weekyear()`



No entanto, se variável de sistema `BrokenWeeks` estiver configurada para usar semanas ininterruptas, a semana 1 deverá conter apenas um determinado número de dias em janeiro com base no valor especificado na variável de sistema `ReferenceDay`.

Por exemplo, se um valor `ReferenceDay` de 4 for usado, a semana 1 deverá incluir pelo menos quatro dias em janeiro. É possível que a semana 1 inclua datas em dezembro do ano anterior ou que o número da última semana de um ano inclua datas em janeiro do ano seguinte. Em situações como essa, a função `weekyear()` retornará um valor diferente para a função `year()`.

Diagrama do intervalo da função `weekyear()` ao usar semanas ininterruptas



Quando usar

A função `weekyear()` é útil quando você deseja comparar agregações por anos. Por exemplo, se você quiser ver o total de vendas de produtos por ano. A função `weekyear()` é escolhida no lugar de `year()` quando o usuário deseja manter a consistência com a variável de sistema `BrokenWeeks` no aplicativo.

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplos de funções

Exemplo	Resultado
`weekyear('12/30/1996',0,0,4)`	Retorna 1997, porque a semana 1 de 1997 começa em 30/12/1996
`weekyear('01/02/1997',0,0,4)`	Retorna 1997

Exemplo	Resultado
`weekyear('12/28/1997',0,0,4)`	Retorna 1997
`weekyear('12/30/1997',0,0,4)`	Retorna 1998, porque a semana 1 de 1998 começa em 29/12/1997
`weekyear('01/02/1999',0,0,4)`	Retorna 1998, porque a semana 53 de 1998 termina em 03/01/1999

Tópicos relacionados

Tópico	Interação
week (page 1060)	Retorna um número inteiro representando o número da semana de acordo com o ISO 8601
year (page 1134)	Retorna um número inteiro que representa o ano em que a expressão é interpretada como uma data, de acordo com a interpretação de número padrão.

Exemplo 1 - Semanas interrompidas

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações da última semana de 2020 e da primeira semana de 2021 que é carregado em uma tabela denominada "`Transactions`",
- A variável `BrokenWeeks` que é definida como 1.
- Um carregamento anterior que contém o seguinte:
  - A função `weekyear()`, definida como o campo "`week_year`", que retorna o ano em que as transações ocorreram.
  - A função `week()`, definida como o campo "`week`", que mostra o número da semana de cada data da transação.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `id`
- `date`
- `week`
- `week_year`

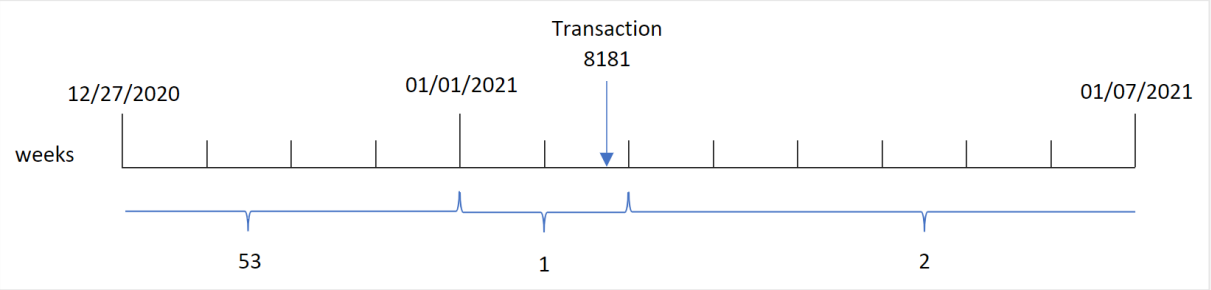
Tabela de resultados

id	date	semana	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

O campo "`week_year`" é criado na instrução de carregamento anterior usando a função `weekyear()` e transmitindo o campo de data como o argumento da função.

A variável de sistema `BrokenWeeks` está definida como `1`, significando que o aplicativo usa semanas interrompidas. A semana 1 começa em 1º de janeiro.

Diagrama do intervalo da função weekyear() com o uso de semanas interrompidas



A transação 8181 ocorre em 2 de janeiro, que faz parte da semana 1. Portanto, ela retorna um valor de 2021 para o campo "week_year".

Exemplo 2 - Semanas ininterruptas

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações da última semana de 2020 e da primeira semana de 2021 que é carregado em uma tabela denominada "Transactions".
- A variável BrokenWeeks, que está definida como 0.
- Um carregamento anterior que contém o seguinte:
  - A função weekyear(), definida como o campo "week_year", que retorna o ano em que as transações ocorreram.
  - A função week(), definida como o campo "week", que mostra o número da semana de cada data da transação.

No entanto, neste exemplo, a política da empresa é usar semanas ininterruptas.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `id`
- `date`
- `week`
- `week_year`

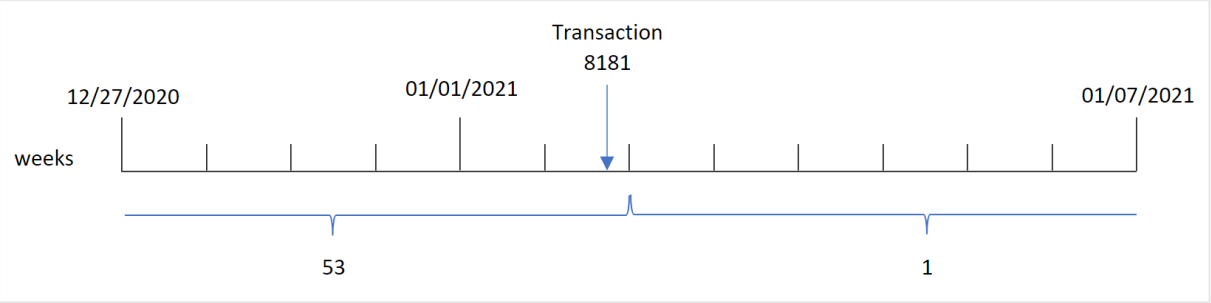
Tabela de resultados

id	date	semana	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

A variável de sistema `BrokenWeeks` está definida como `0`, significando que o aplicativo usa semanas ininterruptas. Portanto, a semana 1 não precisa começar em 1º de janeiro.

A semana 53 de 2020 continua até o final de 2 de janeiro de 2021, com a semana 1 de 2020 começando no domingo, 3 de janeiro de 2021.

Diagrama do intervalo da função weekyear() com o uso de semanas ininterruptas



A transação 8181 ocorre em 2 de janeiro, que faz parte da semana 1. Portanto, ela retorna um valor de 2021 para o campo "week_year".

Exemplo 3: Exemplo de objeto de gráfico

Script de carregamento e expressão de gráfico

Visão geral

São usados o mesmo conjunto de dados e cenário do primeiro exemplo.

No entanto, neste exemplo, o conjunto de dados permanece inalterado e é carregado no aplicativo. O cálculo que retorna o número da semana do ano em que as transações ocorreram é criado como uma medida em um gráfico no aplicativo.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- id
- date

Para calcular a semana em que uma transação ocorre, crie a seguinte medida:

- =week(date)

Para calcular o ano em que uma transação ocorre com base no número da semana, crie a seguinte medida:

- =weekyear(date)

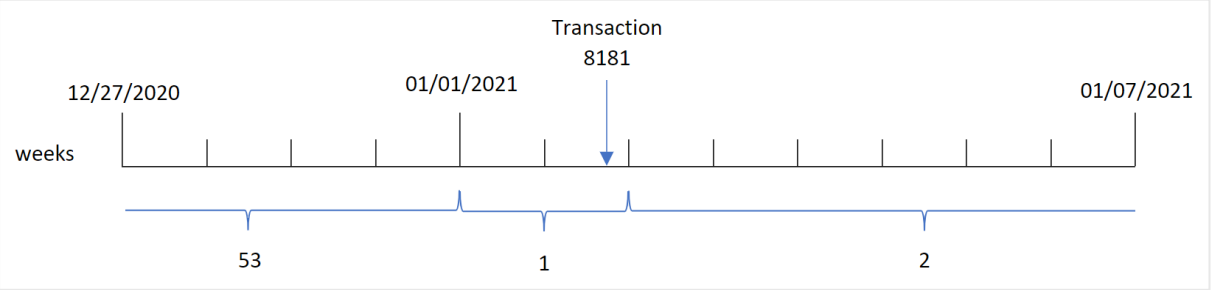
Tabela de resultados

id	date	semana	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

O campo "week_year" é criado na instrução de carregamento anterior usando a função weekyear() e transmitindo o campo de data como o argumento da função.

A variável de sistema BrokenWeeks está definida como 1, significando que o aplicativo usa semanas interrompidas. A semana 1 começa em 1º de janeiro.

Diagrama do intervalo da função weekyear() com o uso de semanas interrompidas



A transação 8181 ocorre em 2 de janeiro, que faz parte da semana 1. Portanto, ela retorna um valor de 2021 para o campo "week_year".

Exemplo 4: Cenário

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações da última semana de 2020 e da primeira semana de 2021 que é carregado em uma tabela denominada "`Transactions`".
- A variável `BrokenWeeks`, que está definida como `0`. Isso significa que o aplicativo usará semanas ininterruptas.
- A variável `ReferenceDay`, que está definida como `2`. Isso significa que o ano começará em 2 de janeiro e conterá no mínimo dois dias em janeiro.
- A variável `FirstWeekDay`, que está definida como `1`. Isso significa que o primeiro dia da semana será terça-feira.

A política da empresa é usar semanas interrompidas. O usuário final gostaria de um gráfico que apresentasse o total de vendas por ano. O aplicativo usa semanas ininterruptas, com a semana 1 contendo no mínimo dois dias em janeiro.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela.

Para calcular o ano em que uma transação ocorre com base no número da semana, crie a seguinte medida:

- `=weekyear(date)`

Para calcular o total de vendas, crie a seguinte medida:

- `sum(amount)`

Defina o **Formato numérico** da medida como **Dinheiro**.

Tabela de resultados

weekyear(date)	=sum(amount)
2020	19.42
2021	373.37

year

Esta função retorna um número inteiro que representa o ano em que a **expression** é interpretada como uma data, de acordo com a interpretação de número padrão.

Sintaxe:

```
year(expression)
```

Tipo de dados de retorno: inteiro

A função `year()` está disponível como função de script e gráfico. A função retorna o ano de uma data específica. Ela é geralmente usada para criar um campo de ano como uma dimensão em um Calendário mestre.

Quando usar

A função `year()` é útil quando você deseja comparar agregações por ano. Por exemplo, a função pode ser usada se você deseja ver o total de vendas de produtos por ano.

Essas dimensões podem ser criadas no script de carregamento usando a função para criar um campo em uma tabela de Calendário mestre. Como alternativa, ela pode ser usada diretamente em um gráfico como uma dimensão calculada.

Exemplos de funções

Exemplo	Resultado
`year( '2012-10-12' )`	retorna 2012
`year( '35648' )`	retorna 1997, por que 35648 = 1997-08-06

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações

regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplo 1 – conjunto de dados DateFormat (script)

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados de datas, que é carregado em uma tabela denominada `Master Calendar`.
- A `variável` de sistema DateFormat padrão (MM/DD/AAAA) é usada.
- Um carregamento anterior, que é usado para criar um campo adicional, `year`, usando a função `year()`.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `year`

Tabela de resultados

date	**ano**
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

Exemplo 2: Datas ANSI

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados de datas, que é carregado em uma tabela denominada `Master Calendar`.
- A variável de sistema `DateFormat` padrão (MM/DD/AAAA) é usada. No entanto, as datas incluídas no conjunto de dados estão no formato de data padrão ANSI.
- Um carregamento anterior, que é usado para criar um campo adicional, denominado `year`, usando a função `year()`.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `year`

Tabela de resultados

date	**ano**
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

date	ano
2021-01-06	2021
2021-01-07	2021

Exemplo 3: Datas não formatadas

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados de datas em formato numérico, que é carregado em uma tabela denominada `Master Calendar`.
- A variável de sistema `DateFormat` padrão (MM/DD/AAAA) é usada.
- Um carregamento anterior, que é usado para criar um campo adicional, `year`, usando a função `year()`.

A data original não formatada é carregada, denominada `unformatted_date`, e, para fornecer clareza, um outro campo adicional, denominado `long_date`, é usado para converter a data numérica em um campo de data formatado usando a função `date()`.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `unformatted_date`
- `long_date`
- `year`

Tabela de resultados

unformatted_date	long_date	ano
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

Exemplo 4: Exemplo de objeto de gráfico

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia.

Neste exemplo, um conjunto de dados de pedidos feitos é carregado em uma tabela denominada Sales. A tabela contém três campos:

- `id`
- `sales_date`
- `amount`

As garantias na venda de produtos duram dois anos a partir da data de venda. A tarefa é criar uma medida em um gráfico para determinar o ano em que cada garantia expirará.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão: `sales_date`.

Crie a seguinte medida:

```
=year(sales_date+365*2)
```

Tabela de resultados

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

Os resultados dessa medida podem ser vistos na tabela acima. Para adicionar dois anos a uma data, multiplique 365 por 2 e adicione o resultado à data da venda. Portanto, as vendas que ocorreram em 2020 têm um ano de expiração de 2022.

yearend

Esta função retorna um valor correspondente a um carimbo de data/hora com o último milissegundo do último dia do ano que contém **date**. O formato de saída padrão será o **DateFormat** definido no script.

Sintaxe:

```
YearEnd( date[, period_no[, first_month_of_year = 1]])
```

Em outras palavras, a função `yearend()` determina em qual ano a data cai. Em seguida, ela retorna um carimbo de data/hora, em formato de data, para o último milissegundo daquele ano. O primeiro mês do ano é, por padrão, janeiro. No entanto, você pode alterar o mês definido como o primeiro usando o argumento `first_month_of_year` na função `yearend()`.

> *A função `yearend()` não considera a `variável` de sistema FirstMonthOfYear. O ano começa em 1º de janeiro, a menos que o argumento `first_month_of_year` seja usado para alterá-lo.*

Diagrama da função `yearend()`.

Quando usar

A função `yearend()` é usada como parte de uma expressão quando você deseja que o cálculo use a fração do ano que ainda não ocorreu. Por exemplo, se você quiser calcular o total de juros ainda não acumulados durante o ano.

Tipo de dados de retorno: dual

Argumentos

Argumento	Descrição
date	A data ou o carimbo de data/hora a ser avaliado.
period_no	**period_no** é um inteiro, em que o valor 0 indica o ano que contém a **date**. Valores negativos em **period_no** indicam anos precedentes e valores positivos indicam anos sucessivos.
first_month_of_year	Se desejar trabalhar com anos (fiscais) que não comecem em janeiro, indique um valor entre 2 e 12 em **first_month_of_year**.

Você pode usar os seguintes valores para definir o primeiro mês do ano no argumento `first_month_of_year`:

Valores first_month_of_year

Month	Valor
Fevereiro	2
Março	3
Abril	4
Maio	5
Junho	6
Julho	7
Agosto	8
Setembro	9
Outubro	10
Novembro	11
Dezembro	12

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplos de funções

Exemplo	Resultado
`yearend('10/19/2001')`	Retorna 12/31/2001 23:59:59.
`yearend('10/19/2001', -1)`	Retorna 12/31/2000 23:59:59.
`yearend('10/19/2001', 0, 4)`	Retorna 03/31/2002 23:59:59.

Exemplo 1: Sem argumentos adicionais

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações entre 2020 e 2022 é carregado em uma tabela chamada "`Transactions`".
- O campo de data foi fornecido no formato da variável de sistema `DateFormat (MM/DD/YYYY)`.
- Uma instrução de carregamento anterior que contém o seguinte:
    - Função `yearend()`, definida como o campo `year_end`.
    - Função `Timestamp()`, definida como o campo `year_end_timestamp`.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- id
- date
- year_end
- year_end_timestamp

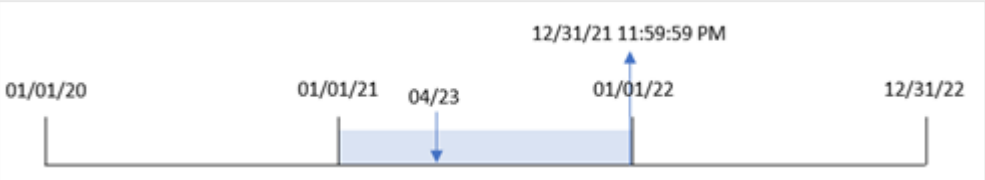
Tabela de resultados

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

O campo "year_end" é criado na instrução de carregamento anterior usando a função yearend() e transmitindo o campo de data como argumento da função.

A função yearend() identifica inicialmente em qual ano o valor da data cai e retorna um carimbo de data/hora para o último milissegundo daquele ano.

Diagrama da função yearend() com a transação 8199 selecionada.



A transação 8199 ocorreu em 23 de abril de 2021. A função yearend() retorna o último milissegundo daquele ano, que é 31 de dezembro às 23:59:59.

Exemplo 2: period_no

Script de carregamento e resultados

Visão geral

São usados o mesmo conjunto de dados e cenário do primeiro exemplo.

No entanto, neste exemplo, a tarefa é criar um campo, "previous_year_end", que retorna o carimbo de data/hora de término do ano anterior ao ano em que uma transação ocorreu.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `id`
- `date`
- `previous_year_end`
- `previous_ year_end_timestamp`

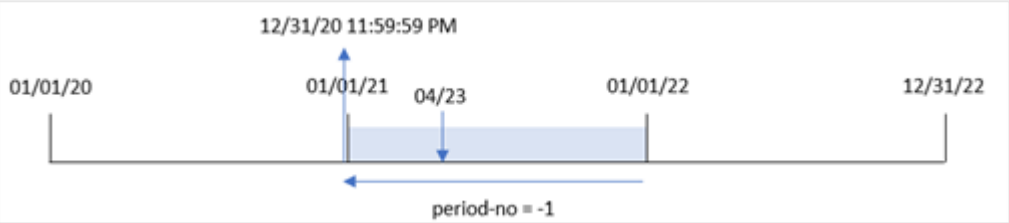
Tabela de resultados

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

id	date	previous_year_end	previous_year_end_timestamp
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

Como um `period_no` de `-1` foi usado como o argumento de deslocamento na função `yearend()`, a função primeiro identifica o ano em que as transações ocorrem. Em seguida, ela procura um ano antes e identifica o último milissegundo daquele ano.

Diagrama da função `yearend()` com period_no de -1.



A transação 8199 ocorre em 23 de abril de 2021. A função `yearend()` retorna o último milissegundo do ano anterior, 31 de dezembro de 2020 às 23:59:59, para o campo "`previous_year_end`".

Exemplo 3: first_month_of_year

Script de carregamento e resultados

Visão geral

São usados o mesmo conjunto de dados e cenário do primeiro exemplo.

No entanto, neste exemplo, a política da empresa é para o ano começar a partir de 1º de abril.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `id`
- `date`
- `year_end`
- `year_end_timestamp`

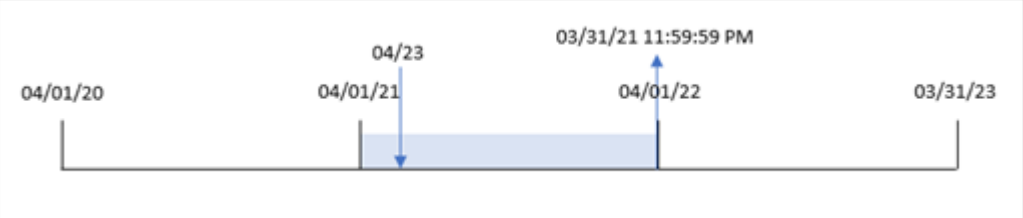
Tabela de resultados

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

id	date	year_end	year_end_timestamp
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

Como o argumento `first_month_of_year` de `4` é usado na função `yearend()`, ele define o primeiro dia do ano como 1º de abril e o último dia do ano como 31 de março.

Diagrama da função `yearend()` com abril como o primeiro mês do ano.



A transação 8199 ocorre em 23 de abril de 2021. Como a função `yearend()` define o início do ano como 1º de abril, ela retorna 31 de março de 2022 como o valor "year_end" para a transação.

Exemplo 4: Exemplo de objeto de gráfico

Script de carregamento e expressão de gráfico

Visão geral

São usados o mesmo conjunto de dados e cenário do primeiro exemplo.

No entanto, neste exemplo, o conjunto de dados permanece inalterado e é carregado no aplicativo. O cálculo que retorna o carimbo de data/hora de término do ano em que uma transação ocorreu é criado como uma medida em um objeto de gráfico do aplicativo.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `id`
- `date`

Para calcular em qual ano uma transação ocorreu, crie as seguintes medidas:

- `=yearend(date)`
- `=timestamp(yearend(date))`

Tabela de resultados

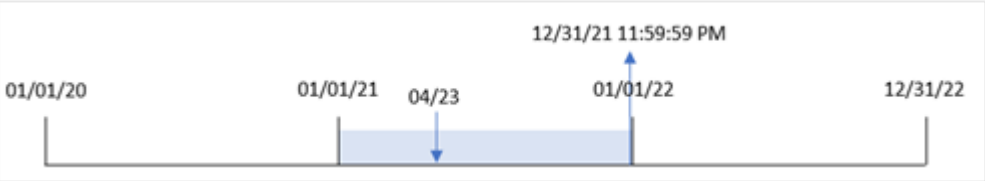
id	date	=yearend(date)	=timestamp(yearend(date))
8188	01/13/2020	12/31/2020	12/31/2020 11:59:59 PM
8189	02/26/2020	12/31/2020	12/31/2020 11:59:59 PM
8190	03/27/2020	12/31/2020	12/31/2020 11:59:59 PM
8191	04/16/2020	12/31/2020	12/31/2020 11:59:59 PM
8192	05/21/2020	12/31/2020	12/31/2020 11:59:59 PM
8193	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

id	date	=yearend(date)	=timestamp(yearend(date))
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

A medida "end_of_year" é criada no objeto de gráfico usando a função `yearend()` e transmitindo o campo de data como argumento da função.

A função `yearend()` identifica inicialmente em qual ano o valor da data cai, retornando um carimbo de data/hora para o último milissegundo daquele ano.

Diagrama da função `yearend()` mostrando que a transação 8199 ocorreu em abril.



A transação 8199 ocorre em 23 de abril de 2021. A função `yearend()` retorna o último milissegundo daquele ano, que é 31 de dezembro às 23:59:59.

Exemplo 5: Cenário

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados é carregado em uma tabela chamada "`Employee_Expenses`". A tabela contém os seguintes campos:
  - IDs de funcionários
  - nome do funcionário
  - reivindicações de despesas médias diárias de cada funcionário

O usuário final deseja um objeto de gráfico que mostre, por ID de funcionário e nome de funcionário, as reivindicações de despesas estimadas ainda a serem acumuladas para o restante do ano. O exercício financeiro começa em janeiro.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `employee_id`
- `employee_name`

Para calcular as reivindicações de despesas projetadas, crie a seguinte medida:

`=(yearend(today(1))-today(1))*avg_daily_claim`
Defina o **Formato numérico** da medida como **Dinheiro**.

Tabela de resultados

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

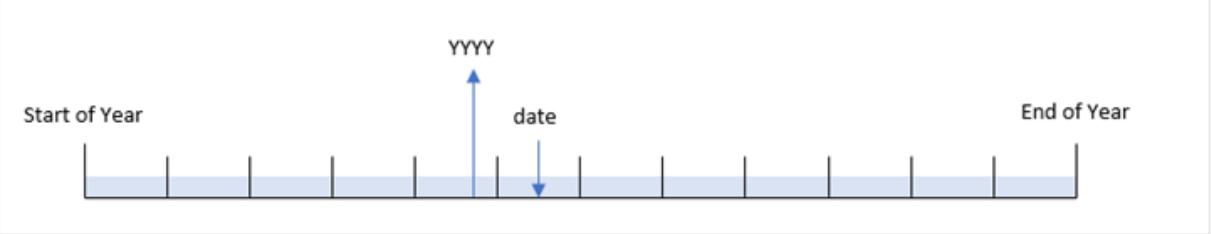
Usando a data de hoje como o único argumento, a função `yearend()` retorna a data de término do ano atual. Em seguida, subtraindo a data de hoje da data de término do ano, a expressão retorna o número de dias restantes naquele ano.

Esse valor é então multiplicado pela média de solicitações de despesas diárias por cada funcionário para calcular o valor estimado das solicitações que cada funcionário deve fazer no ano restante.

yearname

Esta função retorna um ano com quatro dígitos como valor de exibição com um valor numérico subjacente que corresponde a um carimbo de data/hora com o primeiro milissegundo do primeiro dia do ano que contém **date**.

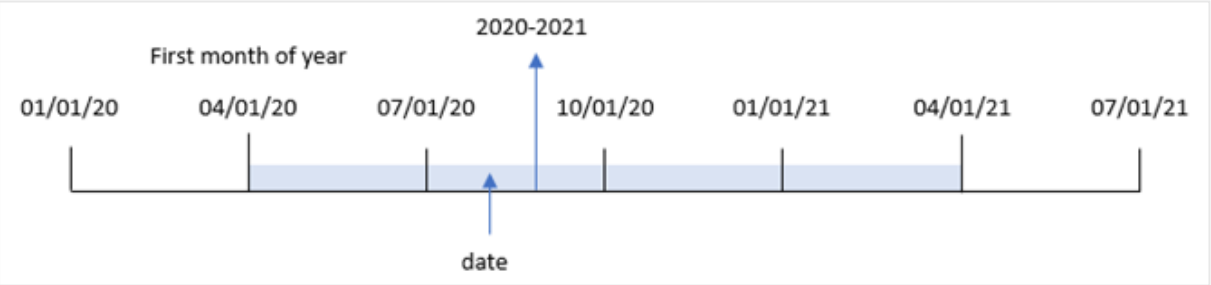
Diagrama do intervalo de tempo da função `yearname()`.



A função `yearname()` é diferente da função `year()`, pois permite deslocar a data que você deseja avaliar e permite definir o primeiro mês do ano.

Se o primeiro mês do ano não for janeiro, a função retornará os dois anos de quatro dígitos durante o período de doze meses que contêm a data. Por exemplo, se o início do ano for abril e a data avaliada for 30/06/2020, o resultado retornado será 2020–2021.

Diagrama da função `yearname()` com abril definido como o primeiro mês do ano.



Sintaxe:

```
YearName(date[, period_no[, first_month_of_year]] )
```

Tipo de dados de retorno: dual

Argumento	Descrição
date	A data ou o carimbo de data/hora a ser avaliado.
period_no	**period_no** é um inteiro, em que o valor 0 indica o ano que contém a **date**. Valores negativos em **period_no** indicam anos precedentes e valores positivos indicam anos sucessivos.
first_month_of_year	Se desejar trabalhar com anos (fiscais) que não comecem em janeiro, indique um valor entre 2 e 12 em **first_month_of_year**. O valor de exibição serão caracteres que mostram dois anos.

Você pode usar os seguintes valores para definir o primeiro mês do ano no argumento `first_month_of_year`:

Valores first_month_of_year

Month	Valor
Fevereiro	2
Março	3
Abril	4
Maio	5
Junho	6
Julho	7
Agosto	8
Setembro	9
Outubro	10
Novembro	11
Dezembro	12

Quando usar

A função `yearname()` é útil para comparar agregações por ano. Por exemplo, se você quiser ver o total de vendas de produtos por ano.

Essas dimensões podem ser criadas no script de carregamento usando a função para criar um campo em uma tabela de Calendário mestre. Elas também podem ser criadas em um gráfico como dimensões calculadas

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplos de funções

Exemplo	Resultado
`yearname('10/19/2001')`	Retorna "2001".
`yearname('10/19/2001',-1)`	Retorna "2000".
`yearname('10/19/2001',0,4)`	Retorna "2001-2002".

Tópicos relacionados

Tópico	Descrição
year (page 1134)	Essa função retorna um número inteiro que representa o ano em que a expressão é interpretada como uma data, de acordo com a interpretação de número padrão.

Exemplo 1: Sem argumentos adicionais

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações entre 2020 e 2022 é carregado em uma tabela chamada "`Transactions`".
- A `variável` de sistema DateFormat que é definida como "`MM/DD/YYYY`".
- Um carregamento anterior que usa o `yearname()` e que está definido como o campo `year_name`.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `year_name`

Tabela de resultados

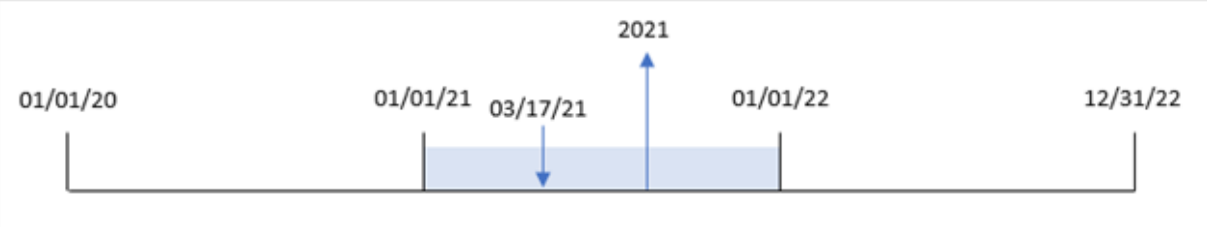
date	year_name
01/13/2020	2020
02/26/2020	2020
03/27/2020	2020
04/16/2020	2020
05/21/2020	2020
08/14/2020	2020
10/07/2020	2020
12/05/2020	2020
01/22/2021	2021
02/03/2021	2021
03/17/2021	2021
04/23/2021	2021
05/04/2021	2021
06/30/2021	2021
07/26/2021	2021
12/27/2021	2021

date	year_name
06/06/2022	2022
07/18/2022	2022
11/14/2022	2022
12/12/2022	2022

O campo "year_name" é criado na instrução de carregamento anterior usando a função yearname() e transmitindo o campo de data como o argumento da função.

A função yearname() identifica em qual ano o valor de data cai e o retorna como um valor de ano de quatro dígitos.

Diagrama da função yearname() que mostra 2021 como o valor do ano.



Exemplo 2: period_no

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações entre 2020 e 2022 é carregado em uma tabela chamada "Transactions".
- A variável de sistema DateFormat , que está definida como "MM/DD/YYYY".
- Um carregamento anterior que usa yearname() e que está definido como o campo year_name.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `prior_year_name`

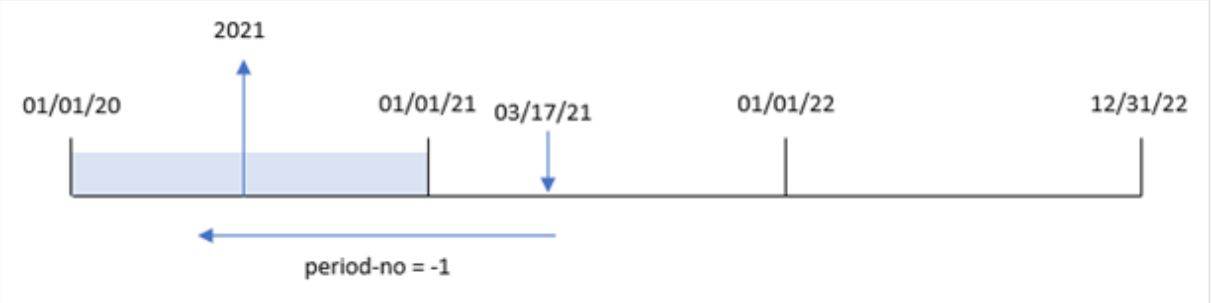
Tabela de resultados

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

date	prior_year_name
04/23/2021	2020
05/04/2021	2020
06/30/2021	2020
07/26/2021	2020
12/27/2021	2020
06/06/2022	2021
07/18/2022	2021
11/14/2022	2021
12/12/2022	2021

Como um `period_no` de `-1` foi usado como o argumento de deslocamento na função `yearname()`, a função primeiro identifica o ano em que as transações ocorrem. Em seguida, a função muda para um ano antes e retorna o ano resultante.

Diagrama da função `yearname()` com `period_no` definido como `-1`.



Exemplo 3: first_month_of_year

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- O mesmo conjunto de dados do primeiro exemplo.
- A variável de sistema `DateFormat` que está definida como "`MM/DD/YYYY`".
- Um carregamento anterior que usa `yearname()` e que está definido como o campo `year_name`.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `id`
- `date`
- `year_name`

Tabela de resultados

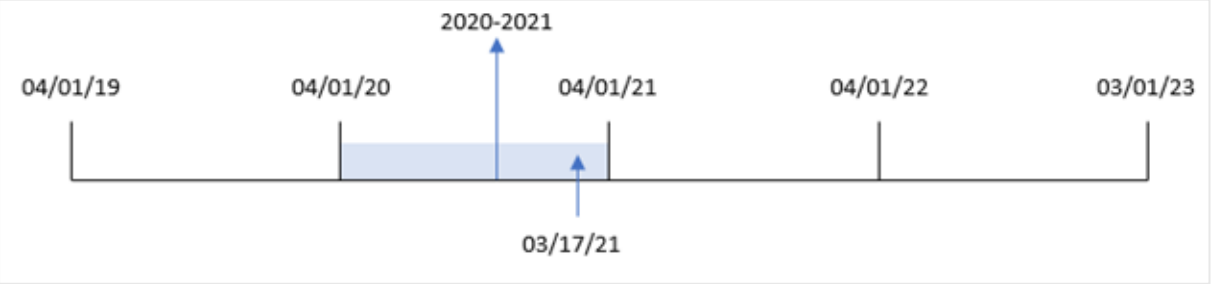
date	year_name
01/13/2020	2019-2020
02/26/2020	2019-2020
03/27/2020	2019-2020
04/16/2020	2020-2021
05/21/2020	2020-2021

date	year_name
08/14/2020	2020-2021
10/07/2020	2020-2021
12/05/2020	2020-2021
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

Como o argumento `first_month_of_year` de `4` é usado na função `yearname()`, o começo do ano passa de 1º de janeiro para 1º de abril. Portanto, cada período de doze meses cruza dois anos civis, e a função `yearname()` retorna os dois anos de quatro dígitos para as datas avaliadas.

A transação 8198 ocorre em 17 de março de 2021. A função `yearname()` define o início do ano em 1º de abril e o final em 30 de março. Portanto, a transação 8198 ocorreu no período do ano de 1º de abril de 2020 e 30 de março de 2021. Como resultado, a função `yearname()` retorna o valor 2020-2021.

Diagrama da função `yearname()` com março definido como o primeiro mês do ano.



Exemplo 4: Exemplo de objeto de gráfico

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- O mesmo conjunto de dados do primeiro exemplo.
- A variável de sistema `DateFormat` que está definida como "`MM/DD/YYYY`".

No entanto, o campo que retorna o ano em que a transação ocorreu é criado como uma medida em um objeto de gráfico.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão:

```
date
```

Para calcular o campo "year_name", crie esta medida:

```
=yearname(date)
```

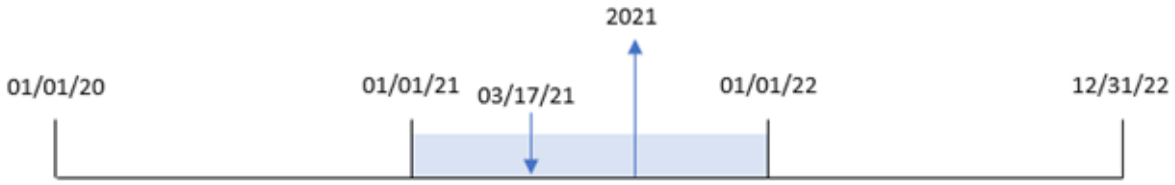
Tabela de resultados

date	=yearname(date)
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

A medida "year_name" é criada no objeto de gráfico usando a função `yearname()` e transmitindo o campo de data como argumento da função.

A função `yearname()` identifica em qual ano o valor de data cai e o retorna como um valor de ano de quatro dígitos.

Diagrama da função yearname() com 2021 como o valor do ano.



Exemplo 5: Cenário

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- O mesmo conjunto de dados do primeiro exemplo.
- A variável de sistema DateFormat que está definida como "MM/DD/YYYY".

O usuário final gostaria de um gráfico que apresentasse o total de vendas por trimestre para as transações. Use a função yearname() como uma dimensão calculada para criar esse gráfico quando a dimensão yearname() não estiver disponível no modelo de dados.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela.

Para comparar agregações por ano, crie esta dimensão calculada:

```
=yearname(date)
```

Crie esta medida:

```
=sum(amount)
```

Defina o **Formato numérico** da medida como **Dinheiro**.

Tabela de resultados

yearname(date)	=sum(amount)
2020	$463.55
2021	$457.69
2022	$294.35

yearstart

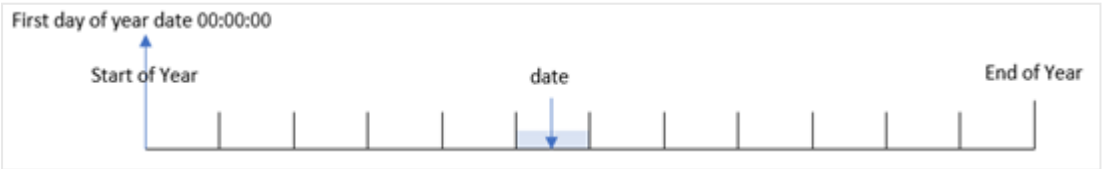
Esta função retorna um carimbo de data/hora correspondente ao início do primeiro dia do ano que contém a **date**. O formato de saída padrão será o **DateFormat** definido no script.

Sintaxe:

```
YearStart(date[, period_no[, first_month_of_year]])
```

Em outras palavras, a função `yearstart()` determina em qual ano a data cai. Em seguida, ela retorna um carimbo de data/hora, em formato de data, para o primeiro milissegundo daquele ano. O primeiro mês do ano é, por padrão, janeiro. No entanto, você pode alterar o mês definido como o primeiro usando o argumento `first_month_of_year` na função `yearstart()`.

Diagrama da função `yearstart()` que mostra o intervalo de tempo que a função pode abranger.



Quando usar

A função `yearstart()` é usada como parte de uma expressão quando você deseja que o cálculo use a fração do ano decorrido até o momento. Por exemplo, se você quiser calcular os juros acumulados em um ano até o momento.

Tipo de dados de retorno: dual

Argumentos

Argumento	Descrição
date	A data ou o carimbo de data/hora a ser avaliado.
period_no	**period_no** é um inteiro, em que o valor 0 indica o ano que contém a **date**. Valores negativos em **period_no** indicam anos precedentes e valores positivos indicam anos sucessivos.
first_month_of_year	Se desejar trabalhar com anos (fiscais) que não comecem em janeiro, indique um valor entre 2 e 12 em **first_month_of_year**.

Os meses seguintes podem ser usados no `first_month_of_year argument`:

Valores first_month_of_year

Month	Valor
Fevereiro	2
Março	3
Abril	4
Maio	5
Junho	6
Julho	7
Agosto	8
Setembro	9
Outubro	10
Novembro	11
Dezembro	12

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações

regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplos de funções

Exemplo	**Resultado**
`yearstart('10/19/2001')`	Retorna 01/01/2001 00:00:00.
`yearstart('10/19/2001',-1)`	Retorna 01/01/2000 00:00:00.
`yearstart('10/19/2001',0,4)`	Retorna 04/01/2001 00:00:00.

Exemplo 1: Exemplo básico

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações entre 2020 e 2022 é carregado em uma tabela chamada "`Transactions`".
- O campo de data foi fornecido no formato da variável de sistema `DateFormat` (`MM/DD/YYYY`).
- Uma instrução de carregamento anterior que contém o seguinte:
    - Função `yearstart()`, definida como o campo `year_start`.
    - Função `Timestamp()`, definida como o campo `year_start_timestamp`

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `id`
- `date`
- `year_start`
- `year_start_timestamp`

Tabela de resultados

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

id	date	year_start	year_start_timestamp
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
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

O campo "year_start" é criado na instrução de carregamento anterior usando a função `yearstart()` e transmitindo o campo de data como o argumento da função.

A função `yearstart()` identifica inicialmente em qual ano o valor da data cai e retorna um carimbo de data/hora para o primeiro milissegundo daquele ano.

Diagrama da função `yearstart()` e a transação 8199.

A transação 8199 ocorreu em 23 de abril de 2021. A função `yearstart()` retorna o primeiro milissegundo daquele ano, que é 1º de janeiro às 12:00:00 AM.

Exemplo 2: period_no

Script de carregamento e resultados

Visão geral

São usados o mesmo conjunto de dados e cenário do primeiro exemplo.

No entanto, neste exemplo, a tarefa é criar um campo, "previous_year_start", que retorna o carimbo de data/hora de início do ano anterior ao ano em que uma transação ocorreu.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- id
- date
- previous_year_start
- previous_ year_start_timestamp

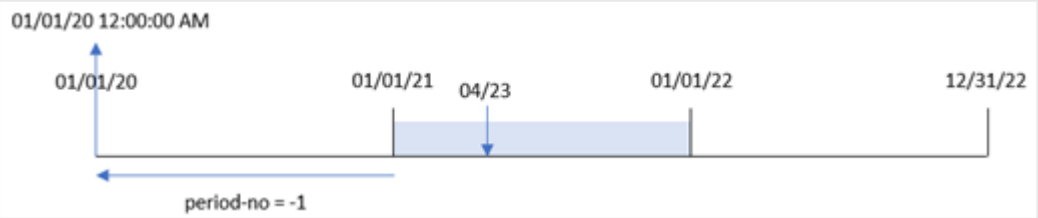
Tabela de resultados

id	date	previous_year_start	previous_year_start_timestamp
8188	01/13/2020	01/01/2019	1/1/2019 12:00:00 AM

id	date	previous_year_start	previous_year_start_timestamp
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
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

Nesse caso, como um `period_no` de `-1` foi usado como o argumento de deslocamento na função `yearstart()`, a função primeiro identifica o ano em que as transações ocorrem. Em seguida, ela procura um ano antes e identifica o primeiro milissegundo daquele ano.

Diagrama da função`yearstart()` com um `period_no` de -1.



A transação 8199 ocorreu em 23 de abril de 2021. A função `yearstart()` retorna o primeiro milissegundo do ano anterior, 1º de janeiro de 2020 às 12:00:00, para o campo "`previous_year_start`".

Exemplo 3: first_month_of_year

Script de carregamento e resultados

Visão geral

São usados o mesmo conjunto de dados e cenário do primeiro exemplo.

No entanto, neste exemplo, a política da empresa é para o ano começar a partir de 1º de abril.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

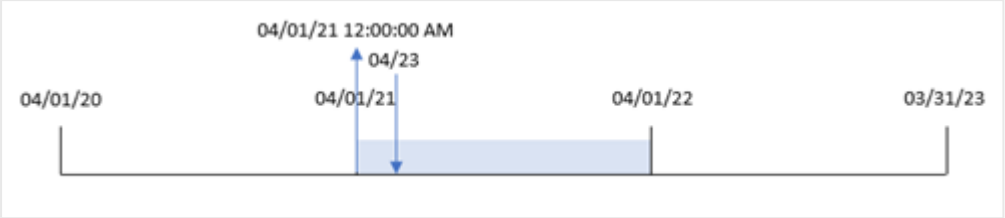
- id
- date
- year_start
- year_start_timestamp

Tabela de resultados

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
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

Nesse caso, como o argumento `first_month_of_year` de `4` é usado na função `yearstart()`, ele define o primeiro dia do ano como 1º de abril e o último dia do ano como 31 de março.

Diagrama da função `yearstart()` com o primeiro mês definido como abril.



A transação 8199 ocorreu em 23 de abril de 2021. Como a função `yearstart()` define o início do ano como 1º de abril, ela o retorna como o valor "`year_start`" para a transação.

Exemplo 4: Exemplo de objeto de gráfico

Script de carregamento e expressão de gráfico

Visão geral

São usados o mesmo conjunto de dados e cenário do primeiro exemplo.

No entanto, neste exemplo, o conjunto de dados permanece inalterado e é carregado no aplicativo. O cálculo que retorna o carimbo de data/hora de início do ano em que uma transação ocorreu é criado como uma medida em um objeto de gráfico do aplicativo.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `id`
- `date`

Para calcular em qual ano uma transação ocorreu, crie as seguintes medidas:

- `=yearstart(date)`
- `=timestamp(yearstart(date))`

Tabela de resultados

id	date	=yearstart(date)	=timestamp(yearstart(date))
8188	06/06/2022	01/01/2022	1/1/2022 12:00:00 AM
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

A medida "`start_of_year`" é criada no objeto de gráfico usando a função `yearstart()` e transmitindo o campo de data como argumento da função.

A função yearstart() identifica inicialmente em qual ano o valor da data cai e retorna um carimbo de data/hora para o primeiro milissegundo daquele ano.

Diagrama da função yearstart() e a transação 8199.

A transação 8199 ocorreu em 23 de abril de 2021. A função yearstart() retorna o primeiro milissegundo daquele ano, que é 1º de janeiro às 12:00:00 AM.

Exemplo 5: Cenário

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados é carregado em uma tabela chamada "Loans". A tabela contém os seguintes campos:
    - IDs de empréstimo.
    - O saldo no início do ano.
    - A taxa de juros simples cobrada em cada empréstimo por ano.

O usuário final deseja um objeto de gráfico que mostre, por ID de empréstimo, os juros atuais que foram acumulados em cada empréstimo no ano até o momento.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- loan_id
- start_balance

Para calcular os juros acumulados, crie a seguinte medida:

```
=start_balance*(rate*(today(1)-yearstart(today(1)))/365)
```

Defina o **Formato numérico** da medida como **Dinheiro**.

Tabela de resultados

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

A função `yearstart()`, usando a data de hoje como seu único argumento, retorna a data de início do ano atual. Ao subtrair esse resultado da data atual, a expressão retorna o número de dias decorridos até o momento neste ano.

Esse valor é então multiplicado pela taxa de juros e dividido por 365 para retornar a taxa de juros efetiva do período. A taxa de juros efetiva do período é então multiplicada pelo saldo inicial do empréstimo para retornar os juros que foram acumulados neste ano até o momento.

yeartodate

Esta função descobre se o carimbo de data e hora de entrada cai dentro de um ano da data em que o script foi carregado pela última vez, e retorna True se cair, False se não cair.

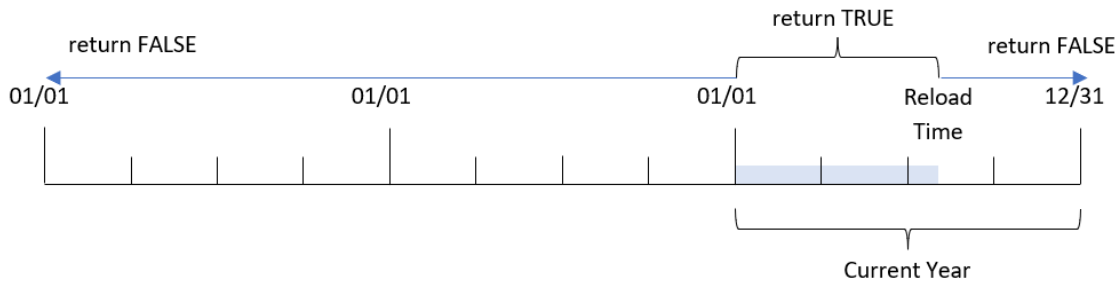
Sintaxe:

```
YearToDate(timestamp[ , yearoffset [ , firstmonth [ , todaydate] ] ])
```

Tipo de dados de retorno: Booleano

No Qlik Sense, o valor booleano "true" é representado por -1, e o valor falso é representado por 0.

Exemplo de diagrama da função `yeartodate()`



Se nenhum dos parâmetros opcionais for usado, o acumulado no ano será qualquer data que ocorrer dentro de um ano civil, a partir de 1º de janeiro até a data da última execução do script, inclusive.

Em outras palavras, a função `yeartodate()`, quando acionada sem parâmetros adicionais, é usada para avaliar um carimbo de data/hora e retornar um resultado booleano com base no fato de a data ter ocorrido no ano civil até a data em que o carregamento ocorreu.

No entanto, também é possível substituir a data de início do ano usando o argumento `firstmonth`, bem como fazer comparações com anos anteriores ou seguintes usando o argumento `yearoffset`.

Finalmente, em casos de conjuntos de dados históricos, a função `yeartodate()` fornece um parâmetro a ser definido como `todaydate`, que comparará o carimbo de data/hora com o ano civil até e incluindo a data fornecida no argumento `todaydate`.

Argumentos

Argumento	Descrição
timestamp	O carimbo de data/hora a ser avaliado, por exemplo, "10/12/2012".
yearoffset	Ao especificar um **yearoffset**, **yeartodate** retorna True para o mesmo período em outro ano. Um **yearoffset** negativo indica um ano anterior e um offset positivo um ano futuro. O year-to-date mais recente é alcançado ao especificar yearoffset = -1. Se omitido, o valor 0 é assumido.
firstmonth	Ao especificar um **firstmonth** entre 1 e 12 (1, se omitido), o início do ano pode ser movido para o primeiro dia de qualquer mês. Por exemplo, para trabalhar com um ano fiscal que inicia em 1º de maio, especifique **firstmonth** = 5. Um valor de 1 indicaria um ano fiscal começando em 1º de janeiro, e um valor de 12 indicaria um ano fiscal começando em 1º de dezembro.
todaydate	Ao especificar um **todaydate** (carimbo de data/hora da execução do último script, se omitido), é possível mover o dia usado como o limite superior do período.

Quando usar

A função `yeartodate()` retorna um resultado booleano. Normalmente, esse tipo de função será usado como uma condição em uma expressão If. Isso retornaria uma agregação ou cálculo dependendo se a data avaliada ocorreu no ano até e incluindo a data da última recarga do aplicativo.

Por exemplo, a função YearToDate() pode ser usada para identificar todos os equipamentos fabricados até o momento no ano atual.

Os exemplos a seguir presumem que o tempo do último carregamento é de 18/11/2011.

Exemplos de funções

Exemplo	**Resultado**
`yeartodate( '11/18/2010')`	retorna False
`yeartodate( '02/01/2011')`	retorna True
`yeartodate( '11/18/2011')`	retorna True
`yeartodate( '11/19/2011')`	retorna False
`yeartodate( '11/19/2011', 0, 1, '12/31/2011')`	retorna True
`yeartodate( '11/18/2010', -1)`	retorna True
`yeartodate( '11/18/2011', -1)`	retorna False
`yeartodate( '04/30/2011', 0, 5)`	retorna False
`yeartodate( '05/01/2011', 0, 5)`	retorna True

Configurações regionais

A menos que especificado de outra forma, os exemplos neste tópico usam o seguinte formato de data: MM/DD/AAAA. O formato de data é especificado na instrução `SET DateFormat` no seu script de carregamento de dados. A formatação de data padrão pode ser diferente no seu sistema devido às suas configurações regionais e outros fatores. Você pode alterar os formatos nos exemplos abaixo para atender às suas necessidades. Ou pode alterar os formatos no seu script de carregamento para corresponder a esses exemplos.

As configurações regionais padrão nos aplicativos são baseadas nas configurações regionais do sistema do computador ou servidor em que o Qlik Sense está instalado. Se o servidor Qlik Sense que você está acessando estiver definido como Suécia, o Editor de carregamento de dados usará as configurações regionais suecas para datas, horas e moedas. Essas configurações de formato regional não estão relacionadas ao idioma exibido na interface do usuário do Qlik Sense. O Qlik Sense será exibido no mesmo idioma do navegador que você está usando.

Exemplo 1: Exemplo básico

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações entre 2020 e 2022, que é carregado em uma tabela denominada `Transactions`.
- O campo de data fornecido no formato da variável de sistema `DateFormat` (MM/DD/AAAA).
- A criação de um campo `year_to_date`, que determina quais transações ocorreram no ano civil até a data do último carregamento.

No momento em que este artigo foi escrito, a data é 26 de abril de 2022.

Script de carregamento

```
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
];
```

Resultados

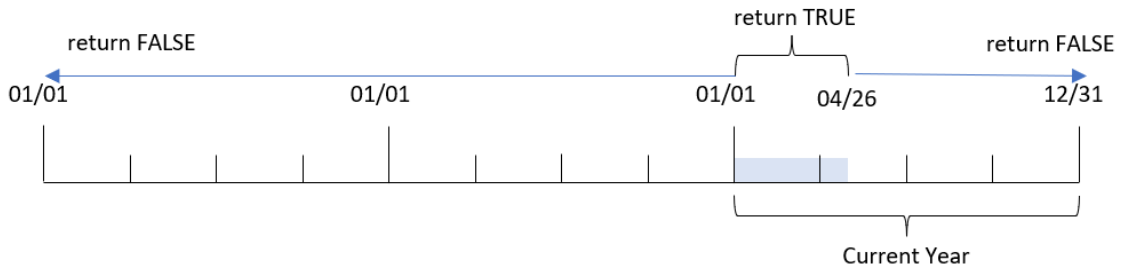
Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `year_to_date`

Tabela de resultados

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
07/26/2021	0
12/27/2021	0
02/02/2022	-1
02/26/2022	-1
03/07/2022	-1
03/11/2022	-1

Diagrama da função `yeartodate()`, exemplo básico



O campo `year_to_date` é criado na instrução de carregamento anterior usando a função `yeartodate()` e transmitindo o campo `date` como o argumento da função.

Como nenhum outro parâmetro é informado para a função, a função `yeartodate()` identifica inicialmente a data de carregamento e, portanto, os limites do ano civil atual (a partir de 1º de janeiro) que retornará um resultado booleano de TRUE.

Portanto, qualquer transação que ocorra entre 1º de janeiro e 26 de abril, a data de carregamento, retornará um resultado booleano de TRUE. Qualquer transação que ocorra antes do início de 2022 retornará um resultado booleano de FALSE.

Exemplo 2: yearoffset

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- O mesmo conjunto de dados e cenário do primeiro exemplo.
- A criação de um campo, `two_years_prior`, que determina quais transações ocorreram dois anos antes do ano civil até o momento.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

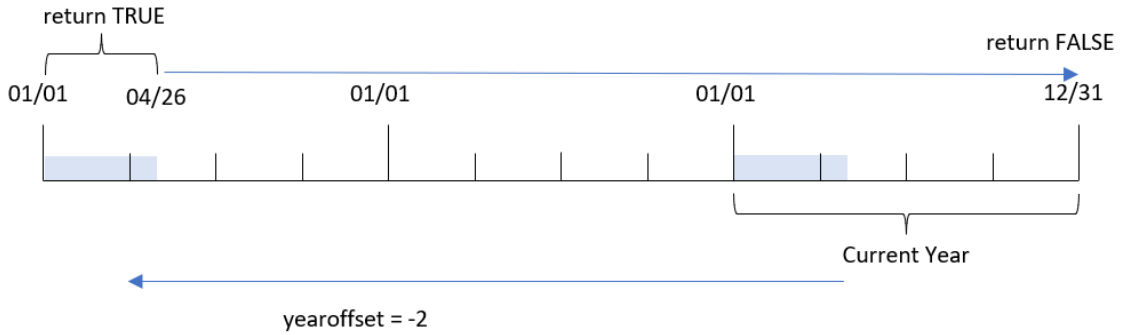
- `date`
- `two_years_prior`

Tabela de resultados

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

Ao usar -2 como argumento `yearoffset` na função `yeartodate()`, a função altera os limites do segmento comparador do ano civil em dois anos completos. Inicialmente, o segmento do ano equivale a entre 1º de janeiro e 26 de abril de 2022. O argumento `yearoffset` então compensa esse segmento com dois anos antes. Os limites de data cairão então entre 1º de janeiro e 26 de abril de 2020.

Diagrama da função `yeartodate()`, exemplo de yearoffset



Portanto, qualquer transação que ocorra entre 1º de janeiro e 26 de abril de 2020 retornará um resultado booleano de TRUE. Todas as transações que aparecerem antes ou depois desse segmento retornarão FALSE.

Exemplo 3: firstmonth

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- O mesmo conjunto de dados e cenário do primeiro exemplo.
- A criação de um campo `year_to_date`, que determina quais transações ocorreram no ano civil até a data do último carregamento.

Neste exemplo, definimos o início do ano fiscal como 1º de julho.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `year_to_date`

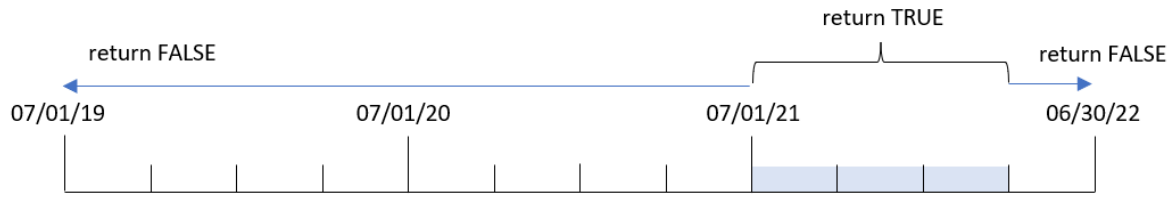
Tabela de resultados

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

date	year_to_date
04/23/2021	0
05/04/2021	0
06/30/2021	0
07/26/2021	-1
12/27/2021	-1
02/02/2022	-1
02/26/2022	-1
03/07/2022	-1
03/11/2022	-1

Nesse caso, como o argumento `firstmonth` de 7 é usado na função `yeartodate()`, ele define o primeiro dia do ano como 1º de julho e o último dia do ano como 30 de junho.

Diagrama da função `yeartodate()`, exemplo de firstmonth



Portanto, qualquer transação que ocorra entre 1º de julho de 2021 e 26 de abril de 2022, a data de carregamento, retornará um resultado booleano de TRUE. Qualquer transação que ocorra antes de 1º de julho de 2021 retornará um resultado booleano de FALSE.

Exemplo 4: todaydate

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- O mesmo conjunto de dados e cenário do primeiro exemplo.
- A criação de um campo `year_to_date`, que determina quais transações ocorreram no ano civil até a data do último carregamento.

No entanto, neste exemplo, precisamos identificar todas as transações que ocorreram no ano civil até 1º de março de 2022, inclusive.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `date`
- `year_to_date`

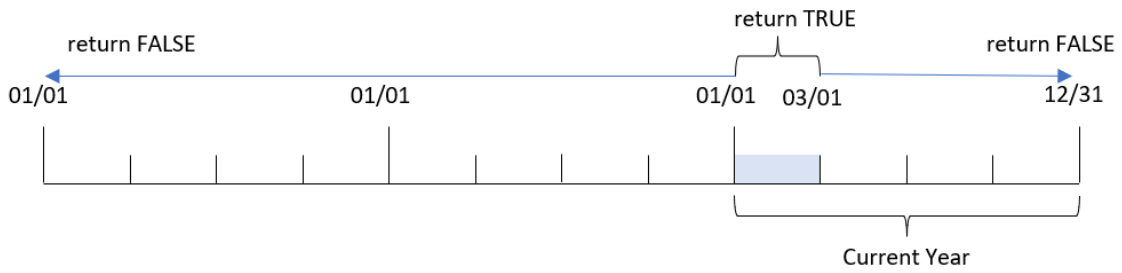
Tabela de resultados

date	year_to_date
01/10/2020	0
02/28/2020	0

date	year_to_date
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	0
03/11/2022	0

Nesse caso, como o argumento `todaydate` de 03/01/2022 é usado na função `yeartodate()`, ele define o limite final do segmento do ano civil comparador como 1º de março de 2022. É fundamental fornecer o parâmetro `firstmonth` (entre 1 e 12); caso contrário, a função retornará resultados nulos.

Diagrama da função `yeartodate()`, exemplo usando o argumento todaydate



Portanto, qualquer transação que ocorra entre 1º de janeiro de 2022 e 1º de março de 2022, o parâmetro `todaydate`, retornará um resultado booleano de TRUE. Qualquer transação que ocorra antes de 1º de janeiro de 2022 ou depois de 1º de março de 2022 retornará um resultado booleano de FALSE.

Exemplo 5: Exemplo de objeto de gráfico

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor de carregamento de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém o mesmo conjunto de dados e cenário do primeiro exemplo.

No entanto, neste exemplo, o conjunto de dados inalterado é carregado no aplicativo. O cálculo que determina quais transações ocorreram no ano civil até a data do último carregamento é criado como uma medida em um objeto de gráfico do aplicativo.

Script de carregamento

```
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
];
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esse campo como uma dimensão: `date`.

Adicione a seguinte medida:

```
=yeartodate(date)
```

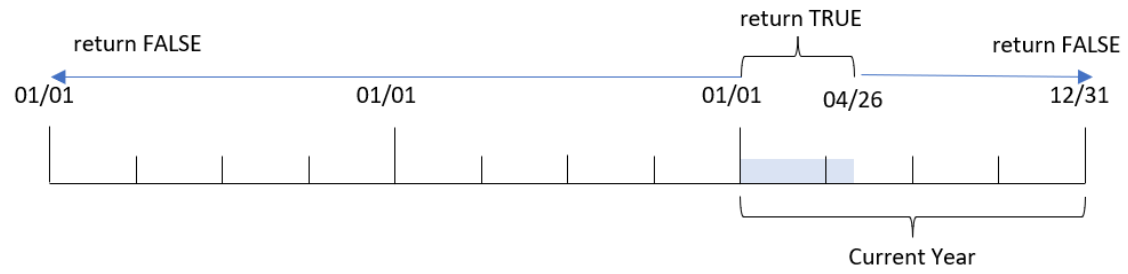
Tabela de resultados

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

A medida `year_to_date` é criada no objeto de gráfico usando a função `yeartodate()` e transmitindo o campo `date` como o argumento da função.

Como nenhum outro parâmetro é informado para a função, a função `yeartodate()` identifica inicialmente a data de carregamento e, portanto, os limites do ano civil atual (a partir de 1º de janeiro) que retornará um resultado booleano de TRUE.

Diagrama da função `yeartodate()`, exemplo usando objeto de gráfico



Qualquer transação que ocorra entre 1º de janeiro e 26 de abril, a data de carregamento, retornará um resultado booleano de TRUE. Qualquer transação que ocorra antes do início de 2022 retornará um resultado booleano de FALSE.

Exemplo 6: cenário

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Um conjunto de dados contendo um conjunto de transações entre 2020 e 2022, que é carregado em uma tabela chamada `Transactions`.
- O campo de data fornecido no formato da variável de sistema `DateFormat` (MM/DD/AAAA).

O usuário final gostaria de um objeto de KPI que apresentasse o total de vendas para o período equivalente em 2021 como o ano atual até o momento do último carregamento.

No momento em que este artigo foi escrito, a data é 16 de junho de 2022.

Script de carregamento

```
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
```

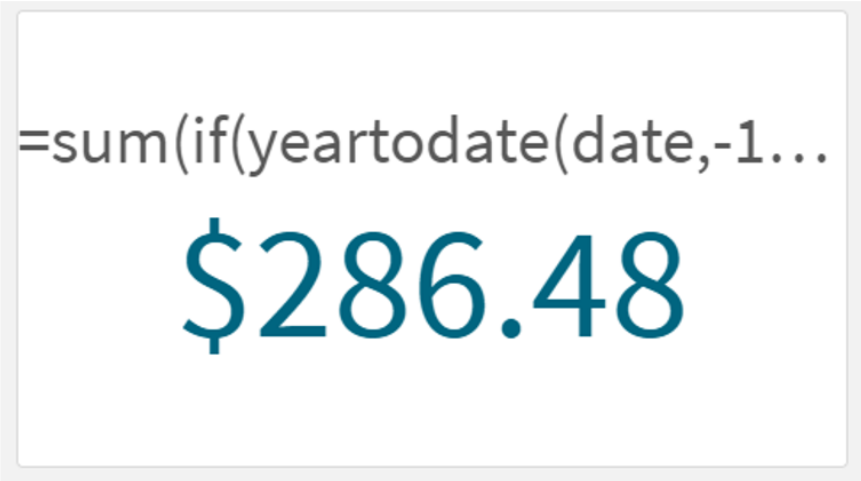
```
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
8200,05/04/2021,82.77
8201,06/30/2021,69.98
8202,07/26/2021,76.11
8203,12/27/2021,25.12
8204,02/02/2022,46.23
8205,02/26/2022,84.21
8206,03/07/2022,96.24
8207,03/11/2022,67.67
];
```

Resultados

Faça o seguinte:

1. Crie um objeto de KPI.
2. Crie a seguinte medida de agregação para calcular o total de vendas:
   `=sum(if(yeartodate(date,-1),amount,0))`
3. Defina o **Formato numérico** da medida como **Dinheiro**.

Gráfico de KPI `yeartodate()` para 2021



A função `yeartodate()` retorna um valor booleano ao avaliar as datas de cada ID de transação. Como o carregamento ocorreu em 16 de junho de 2022, a função `yeartodate` segmenta o período do ano entre 01/01/2022 e 16/06/2022. No entanto, como um valor `period_no` de -1 foi usado na função, esses limites são então transferidos para o ano anterior. Portanto, para qualquer transação que ocorra entre 01/01/2021 e 16/06/2021, a função `yeartodate()` retorna um valor booleano de `TRUE` e soma o valor.

5.8 Funções exponenciais e logarítmicas

Esta seção descreve as funções relacionadas aos cálculos exponenciais e logarítmicos. Todas as funções podem ser usadas no script de carregamento de dados e em expressões de gráfico.

Nas funções a seguir, os parâmetros são expressões em que **x** e **y** deve ser interpretado como números de valor real.

exp
Função exponencial natural, e^x, usando como base o logaritmo natural **e**. O resultado é um número positivo.

```
exp(x )
```

Exemplos e resultados:

`exp(3)` retorna 20,085.

log
O logaritmo natural de **x**. A função somente será definida se **x**> 0. O resultado é um número.

```
log(x )
```

Exemplos e resultados:

`log(3)` retorna 1,0986.

log10
Logaritmo comum (base 10) de **x**. A função somente será definida se **x**> 0. O resultado é um número.

```
log10(x )
```

Exemplos e resultados:

`log10(3)` retorna 0,4771.

pow
Retorna **x** à potência **y**. O resultado é um número.

```
pow(x,y )
```

Exemplos e resultados:

`pow(3, 3)` retorna 27.

sqr
x ao quadrado (**x** na potência de 2). O resultado é um número.

```
sqr (x )
```

Exemplos e resultados:

`sqr(3)` retorna 9.

sqrt
Raiz quadrada de **x**. A função somente será definida se **x** >= 0. O resultado é um número positivo.

```
sqrt(x )
```

Exemplos e resultados:

`sqrt(3)` retorna 1,732.

5.9 Funções de campo

Essas funções só podem ser usadas em expressões de gráficos.

As funções de campo retornam inteiros ou caracteres que identificam os diferentes aspectos das seleções de campo.

Funções de contagem

GetAlternativeCount
GetAlternativeCount() é utilizada para encontrar o número de valores alternativos (cinza-claro) no campo identificado.

```
GetAlternativeCount - função de gráfico (field_name)
```

GetExcludedCount
GetExcludedCount() encontra o número de valores distintos excluídos no campo identificado. Valores excluídos incluem campos alternativos (cinza claro), excluídos (cinza escuro) e excluídos selecionados (cinza escuro com marca de seleção).

```
GetExcludedCount - função de gráfico (page 1198)(field_name)
```

GetNotSelectedCount
Essa função de gráfico retorna o número de valores não selecionados no campo nomeado como **fieldname**. O campo deve estar no modo AND para que essa função seja relevante.

```
GetNotSelectedCount - função de gráfico(fieldname [, includeexcluded=false])
```

GetPossibleCount
GetPossibleCount() é usado para encontrar o número de possíveis valores no campo identificado. Se o campo identificado incluir seleções, os campos selecionados (verde) são contados. Caso contrário, os valores associados (branco) são contados.

```
GetPossibleCount - função de gráfico(field_name)
```

GetSelectedCount

GetSelectedCount() encontra o número de valores (verde) selecionados em um campo.

```
GetSelectedCount - função de gráfico (field_name [, include_excluded])
```

Funções de campo e seleção

GetCurrentSelections

O **GetCurrentSelections()** retorna uma lista das seleções atuais no aplicativo. Se as seleções forem feitas usando caracteres de busca em uma caixa de pesquisa, o **GetCurrentSelections()** retornará os caracteres de busca.

```
GetCurrentSelections - função de gráfico([record_sep [,tag_sep [,value_sep
[,max_values]]]])
```

GetFieldSelections

GetFieldSelections() retorna uma **cadeia de caracteres** com as seleções atuais em um campo.

```
GetFieldSelections - função de gráfico ( field_name [, value_sep [, max_
values]])
```

GetObjectDimension

GetObjectDimension() retorna o nome da dimensão. **Index** é um número inteiro opcional que indica a dimensão que deve ser retornada.

```
GetObjectDimension - função de gráfico ([index])
```

GetObjectField

GetObjectField() retorna o nome da dimensão. **Index** é um número inteiro opcional que indica a dimensão que deve ser retornada.

```
GetObjectField - função de gráfico ([index])
```

GetObjectMeasure

GetObjectMeasure() retorna o nome da medida. **Index** é um número inteiro opcional que indica a medida que deve ser retornada.

```
GetObjectMeasure - função de gráfico ([index])
```

GetAlternativeCount - função de gráfico

GetAlternativeCount() é utilizada para encontrar o número de valores alternativos (cinza-claro) no campo identificado.

Sintaxe:

```
GetAlternativeCount (field_name)
```

Tipo de dados de retorno: inteiro

Argumentos:

Argumentos

Argumento	Descrição
field_name	O campo que contém o intervalo de dados que será medido.

Exemplos e resultados:

O exemplo a seguir usa o **campo** First name carregado em um painel de filtro.

Exemplos e resultados

Exemplos	Resultados
Dado que **John** está selecionado em **First name**. `GetAlternativeCount ([First name])`	4 porque há 4 valores exclusivos e excluídos (cinza) em **First name**.
Dado que **John** e **Peter** estão selecionados. `GetAlternativeCount ([First name])`	3 porque há 3 valores exclusivos e excluídos (cinza) em **First name**.
Dado que nenhum valor está selecionado em **First name**. `GetAlternativeCount ([First name])`	0 porque não há seleções.

Dados usados no exemplo:

```
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 - função de gráfico

O **GetCurrentSelections()** retorna uma lista das seleções atuais no aplicativo. Se as seleções forem feitas usando caracteres de busca em uma caixa de pesquisa, o **GetCurrentSelections()** retornará os caracteres de busca.

Se as opções forem utilizadas, você terá que especificar record_sep. Para especificar uma nova linha, defina **record_sep** como **chr(13)&chr(10)**.

Se todos exceto dois, ou todos exceto um, dos valores forem selecionados, o formato 'NOT x,y' ou 'NOT y' será utilizado, respectivamente. Se você selecionou todos os valores e a contagem de todos os valores for maior que max_values, o texto ALL será retornado.

Sintaxe:

```
GetCurrentSelections ([record_sep [, tag_sep [, value_sep [, max_values [, state_name]]]]])
```

Tipo de dados de retorno: caractere

Argumentos:

Argumentos

Argumentos	Descrição
record_sep	Separador a ser colocado entre registros de campo. O padrão é <CR><LF> que significa uma nova linha.
tag_sep	Separador a ser colocado entre o rótulo do nome do campo e os valores de campo. O padrão é ': '.
value_sep	O separador a ser colocado entre valores de campo. O padrão é ', '.
max_values	O número máximo de valores de campo que serão listados individualmente. Quando um grande número de valores é selecionado, será usado o formato “x valores de y”. O padrão é 6.
state_name	O nome de um estado alternativo que foi escolhido para a visualização específica. Se o argumento **state_name** for usado, apenas as seleções associadas ao nome do estado especificado serão levadas em consideração.

Exemplos e resultados:

O exemplo a seguir usa dois campos carregados de diferentes painéis de filtro, um para o nome de **First name** e um para **Initials**.

Exemplos e resultados

Exemplos	Resultados
Dado que **John** está selecionado em **First name**. `GetCurrentSelections ()`	'First name: John'
Dado que **John** e **Peter** estão selecionados em **First name**. `GetCurrentSelections ()`	'First name: John, Peter'
Dado que **John** e **Peter** estão selecionados **First name** e **JA** is está selecionado em **Initials**. `GetCurrentSelections ()`	'First name: John, Peter Initials: JA'

Exemplos	Resultados
Dado que **John** está selecionado em **First name** e **JA** está selecionado em **Initials**. `GetCurrentSelections ( chr(13)&chr(10) , ' = ' )`	'First name = John Initials = JA'
Dado que você selecionou todos os nomes exceto Sue em **First name** e não há seleções em **Initials**. `GetCurrentSelections (chr(13)&chr(10), '=', ',' ,3)`	'First name=NOT Sue'

Dados usados no exemplo:

```
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 - função de gráfico

GetExcludedCount() encontra o número de valores distintos excluídos no campo identificado. Valores excluídos incluem campos alternativos (cinza claro), excluídos (cinza escuro) e excluídos selecionados (cinza escuro com marca de seleção).

Sintaxe:

```
GetExcludedCount (field_name)
```

Tipo de dados de retorno: caractere

Argumentos:

Argumentos

Argumentos	Descrição
field_name	O campo que contém o intervalo de dados que será medido.

Exemplos e resultados:

O exemplo a seguir usa três campos carregados em diferentes painéis de filtro: um para **First name**, um para **Last name** e outro para **Initials**.

Exemplos e resultados

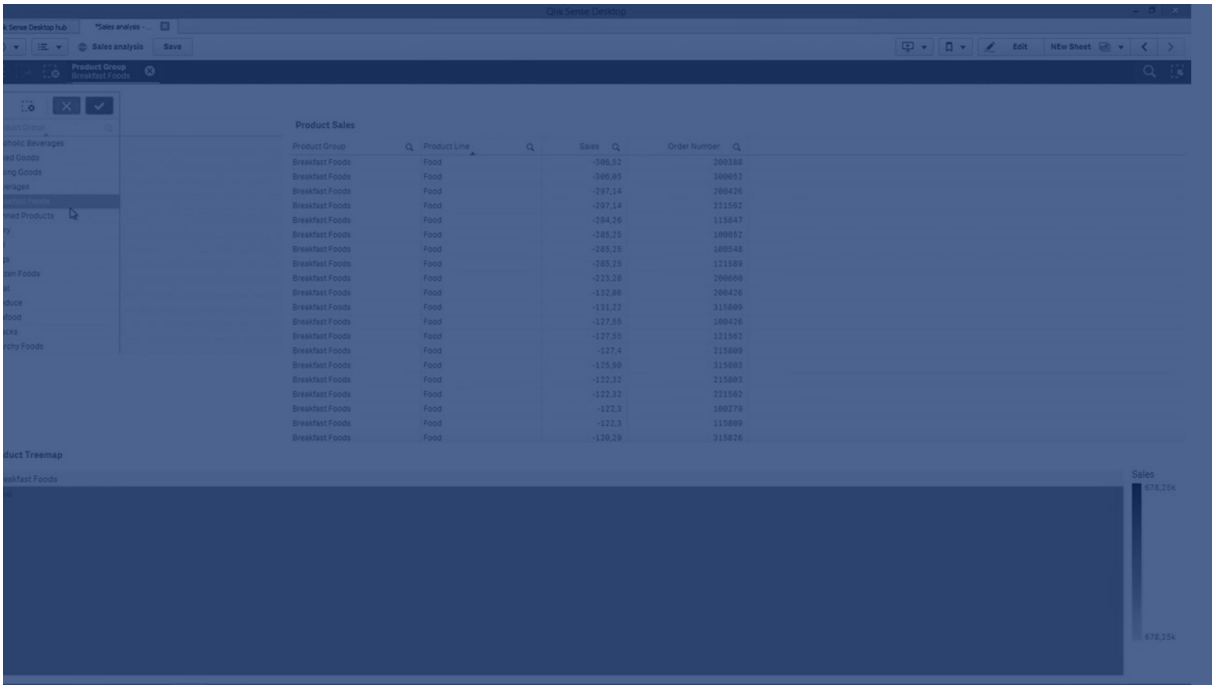
Exemplos	Resultados
Se nenhum valor for selecionado em **First name**.	`GetExcludedCount (Initials) = 0` Não haverá seleções.
Se **John** for selecionado em **First name**.	`GetExcludedCount (Initials) = 5` Haverá 5 valores excluídos em **Iniciais** com a cor cinza escuro. A sexta célula (JA) estará em branco, porque está associada com a seleção John em **First name**.
Se **John** e **Peter** forem selecionados.	`GetExcludedCount (Initials) = 3` John será associado a 1 valor, e Peter será associado a 2 valores em **Initials**.
Se **John** e **Peter** forem selecionados em **First name** e depois **Franc** for selecionado em **Last name**.	`GetExcludedCount ([First name]) = 4` Haverá 4 valores excluídos em **Nome** na cor cinza escuro. **GetExcludedCount()** avalia campos com valores excluídos, incluindo campos excluídos alternativos e selecionados.
Se **John** e **Peter** forem selecionados em **First name** e depois **Franc** e **Anderson** forem selecionados em **Last name**.	`GetExcludedCount (Initials) = 4` Haverá 4 valores excluídos em **Iniciais** com a cor cinza escuro. As outras duas células (JA e PF) serão brancas, pois estão associadas às seleções John e Peter em **First name**.
Se **John** e **Peter** forem selecionados em **First name** e depois **Franc** e **Anderson** forem selecionados em **Last name**.	`GetExcludedCount ([Last name]) = 4` Haverá 4 valores excluídos em **Iniciais**. Devonshire terá a cor cinza claro, enquanto Brown, Carr e Elliot terão a cor cinza escuro.

Dados usados no exemplo:

```
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 - função de gráfico

GetFieldSelections() retorna uma **cadeia de caracteres** com as seleções atuais em um campo.



Se todos exceto dois ou todos exceto um dos valores forem selecionados, o formato 'NOT x,y' ou 'NOT y' será utilizado, respectivamente. Se você selecionou todos os valores e a contagem de todos os valores for maior que max_values, o texto ALL será retornado.

Sintaxe:

```
GetFieldSelections ( field_name [, value_sep [, max_values [, state_name]]])
```

Tipo de dados de retorno: caractere

Formatos de strings de retorno

Formato	Descrição
'a, b, c'	Se o número de valores selecionados for max_values ou menor, a string retornada será uma lista dos valores selecionados. Os valores são separados com value_sep como delimitador.
'NOT a, b, c'	Se o número de valores não selecionados for max_values ou menor, a string retornada será uma lista dos valores não selecionados com NOT como prefixo. Os valores são separados com value_sep como delimitador.
'x of y'	x = o número de valores selecionados y = o número total de valores Isso é retornado quando max_values < x < (y - max_values).
'ALL'	Retornado se todos os valores estiverem selecionados.
'-'	Retornado se nenhum valor estiver selecionado.
<search string>	Se você tiver selecionado com o uso de uma pesquisa, a string de pesquisa será retornada.

Argumentos:

Argumentos

Argumentos	Descrição
field_name	O campo que contém o intervalo de dados que será medido.
value_sep	O separador a ser colocado entre valores de campo. O padrão é ', '.
max_values	O número máximo de valores de campo que serão listados individualmente. Quando um grande número de valores é selecionado, será usado o formato “x valores de y”. O padrão é 6.
state_name	O nome de um estado alternativo que foi escolhido para a visualização específica. Se o argumento **state_name** for usado, apenas as seleções associadas ao nome do estado especificado serão levadas em consideração.

Exemplos e resultados:

O exemplo a seguir usa o **campo** First namecarregado em um painel de filtro.

Exemplos e resultados

Exemplos	Resultados
Dado que **John** está selecionado em **First name**. `GetFieldSelections ([First name])`	'John'
Dado que **John** e **Peter** estão selecionados. `GetFieldSelections ([First name])`	'John,Peter'
Dado que **John** e **Peter** estão selecionados. `GetFieldSelections ([First name],'; ')`	'John; Peter'
Dado que **John**, **Sue**, **Mark** estão selecionados em **First name**. `GetFieldSelections ([First name],';',2)`	'NOT Jane;Peter', pois o valor 2 é indicado como o valor do argumento max_values. Caso contrário, o resultado teria sido John; Sue; Mark.

Dados usados no exemplo:

```
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 - função de gráfico

Essa função de gráfico retorna o número de valores não selecionados no campo nomeado como **fieldname**. O campo deve estar no modo AND para que essa função seja relevante.

Sintaxe:

```
GetNotSelectedCount(fieldname [, includeexcluded=false])
```

Argumentos:

Argumentos

Argumento	Descrição
fieldname	O nome do campo a ser avaliado.
includeexcluded	Se **includeexcluded** for declarado True, a contagem incluirá os valores selecionados que foram excluídos por seleções em outro campo.

Exemplos:

```
GetNotSelectedCount( Country )
```

```
GetNotSelectedCount( Country, true )
```

GetObjectDimension - função de gráfico

GetObjectDimension() retorna o nome da dimensão. **Index** é um número inteiro opcional que indica a dimensão que deve ser retornada.

Não é possível usar essa função em um gráfico nos seguintes locais: título, legenda, rodapé, expressão de linha de referência.

Não é possível fazer referência ao nome de uma dimensão ou medida em outro objeto usando o Object ID.

Sintaxe:

```
GetObjectDimension ([index])
```

Exemplo:

```
GetObjectDimension(1)
```

Exemplo: Expressão de gráfico

Tabela do Qlik Sense mostrando exemplos da função GetObjectDimension em uma expressão de gráfico

transaction_date	ID do cliente	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	ID do cliente

Em vez disso, se quiser retornar o nome de uma medida, use a função **GetObjectMeasure**.

GetObjectField - função de gráfico

GetObjectField() retorna o nome da dimensão. **Index** é um número inteiro opcional que indica a dimensão que deve ser retornada.

Não é possível usar essa função em um gráfico nos seguintes locais: título, legenda, rodapé, expressão de linha de referência.

Não é possível fazer referência ao nome de uma dimensão ou medida em outro objeto usando o Object ID.

Sintaxe:

```
GetObjectField ([index])
```

Exemplo:

```
GetObjectField(1)
```

Exemplo: Expressão de gráfico

Tabela do Qlik Sense mostrando exemplos da função GetObjectField função em uma expressão de gráfico.

transaction_date	ID do cliente	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	ID do cliente

Em vez disso, se quiser retornar o nome de uma medida, use a função **GetObjectMeasure**.

GetObjectMeasure - função de gráfico

GetObjectMeasure() retorna o nome da medida. **Index** é um número inteiro opcional que indica a medida que deve ser retornada.

Não é possível usar essa função em um gráfico nos seguintes locais: título, legenda, rodapé, expressão de linha de referência.

Não é possível fazer referência ao nome de uma dimensão ou medida em outro objeto usando o Object ID.

Sintaxe:

```
GetObjectMeasure ([index])
```

Exemplo:

```
GetObjectMeasure(1)
```

Exemplo: Expressão de gráfico

Tabela do Qlik Sense mostrando exemplos da função GetObjectMeasure em uma expressão de gráfico

ID do cliente	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)

Se você quiser retornar o nome de uma dimensão, use a função **GetObjectField** em vez disso.

GetPossibleCount - função de gráfico

GetPossibleCount() é usado para encontrar o número de possíveis valores no campo identificado. Se o campo identificado incluir seleções, os campos selecionados (verde) são contados. Caso contrário, os valores associados (branco) são contados. .

Para campos com seleções, **GetPossibleCount()** retorna o número de campos selecionados (verde).

Tipo de dados de retorno: inteiro

Sintaxe:

```
GetPossibleCount (field_name)
```

Argumentos:

Argumentos

Argumentos	Descrição
field_name	O campo que contém o intervalo de dados que será medido.

Exemplos e resultados:

O exemplo a seguir usa dois campos carregados de diferentes painéis de filtro, um para o nome de **First name** e um para **Initials**.

Exemplos e resultados

Exemplos	Resultados
Dado que **John** está selecionado em **First name**. `GetPossibleCount ([Initials])`	1 porque há um valor em Initials associado com a seleção, **John**, em **First name**.
Dado que **John** está selecionado em **First name**. `GetPossibleCount ([First name])`	1 porque há uma seleção, **John**, em **First name**.
Dado que **Peter** está selecionado em **First name**. `GetPossibleCount ([Initials])`	2 porque Peter está associado a dois valores em **Initials**.
Dado que nenhum valor está selecionado em **First name**. `GetPossibleCount ([First name])`	5 porque não há seleções e há 5 valores exclusivos em **First name**.
Dado que nenhum valor está selecionado em **First name**. `GetPossibleCount ([Initials])`	6 porque não há seleções e há 6 valores exclusivos em **Initials**.

Dados usados no exemplo:

```
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 - função de gráfico

GetSelectedCount() encontra o número de valores (verde) selecionados em um campo.

Sintaxe:

```
GetSelectedCount (field_name [, include_excluded [, state_name]])
```

Tipo de dados de retorno: inteiro

Argumentos:

Argumentos

Argumentos	Descrição
field_name	O campo que contém o intervalo de dados que será medido.
include_ excluded	Se for definido como **True()**, a contagem incluirá os valores selecionados, atualmente excluídos por seleções em outros campos. Se False ou omitido, esses valores não serão incluídos.
state_name	O nome de um estado alternativo que foi escolhido para a visualização específica. Se o argumento **state_name** for usado, apenas as seleções associadas ao nome do estado especificado serão levadas em consideração.

Exemplos e resultados:

O exemplo a seguir usa três campos carregados de diferentes painéis de filtro, um para nome **First name**, um para **Initials** e um para **Has cellphone**.

Exemplos e resultados

Exemplos	Resultados
Dado que **John** está selecionado em **First name**. `GetSelectedCount ([First name])`	1 porque um valor está selecionado em **First name**.
Dado que **John** está selecionado em **First name**. `GetSelectedCount ([Initials])`	0 porque nenhum valor está selecionado em **Initials**.
Sem nenhuma seleção em **First name**, selecione todos os valores em **Initials** e, em seguida, selecione o valor **Yes** em **Has cellphone**. `GetSelectedCount ([Initials], True())`	6. Embora as seleções com **Initials**MC e PD tenham **Has cellphone** definido como **No**, o resultado ainda será 6, pois o argumento include_excluded está definido como True().

Dados usados no exemplo:

```
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 '|');
```

5.10 Funções de arquivo

As funções do arquivo (disponíveis somente nas expressões de script) retornam informações sobre o arquivo de tabela que está sendo lido no momento. Essas funções retornarão NULL para todas as fontes de dados, exceto para os arquivos de tabela (exceção: **ConnectString( )**).

Visão geral das funções de arquivo

Cada função é descrita adicionalmente após a visão geral. Você também pode clicar no nome da função na sintaxe para acessar imediatamente os detalhes dessa função específica.

Attribute

Essa função de script retorna o valor de meta tags de diferentes arquivos de mídia como texto. Há suporte para os seguintes formatos de arquivo: MP3, WMA, WMV, PNG e JPG. Se o arquivo **filename** não existir, não for um formato de arquivo suportado ou não contiver uma meta tag chamada **attributename**, NULL será retornado.

```
Attribute (filename, attributename)
```

ConnectString

A função **ConnectString()** retorna o nome da conexão de dados ativa para conexões ODBC ou OLE DB . A função retorna uma string vazia se nenhum comando **connect** tiver sido executado ou após uma declaração **disconnect**.

```
ConnectString ()
```

FileBaseName

A função **FileBaseName** retorna uma string contendo o nome do arquivo de tabela que está sendo lido no momento, sem caminho ou extensão.

```
FileBaseName ()
```

FileDir

A função **FileDir** retorna uma string contendo o caminho do diretório do arquivo de tabela que está sendo lido no momento.

```
FileDir ()
```

FileExtension
A função **FileExtension** retorna uma string contendo a extensão do arquivo de tabela que está sendo lido no momento.

```
FileExtension ()
```

FileName
A função **FileName** retorna uma string contendo o nome do arquivo de tabela que está sendo lido no momento, sem caminho, mas incluindo a extensão.

```
FileName ()
```

FilePath
A função **FilePath** retorna uma string contendo o caminho completo do arquivo de tabela que está sendo lido no momento.

```
FilePath ()
```

FileSize
A função **FileSize** retorna um inteiro contendo o tamanho em bytes do arquivo filename ou, se nenhum filename for especificado, retorna o do arquivo de tabela que está sendo lido no momento.

```
FileSize ()
```

FileTime
A função **FileTime** retorna um carimbo de data/hora no formato UTC da última modificação de um arquivo especificado. Se um arquivo não for especificado, a função retornará um carimbo de data/hora em UTC da última modificação do arquivo de tabela atualmente lido.

```
FileTime ([ filename ])
```

GetFolderPath
A função **GetFolderPath** retorna o valor da função do Microsoft Windows *SHGetFolderPath*. Esta função admite como entrada o nome de uma pasta doMicrosoft Windows e retorna o caminho completo da pasta.

```
GetFolderPath ()
```

QvdCreateTime
Esta função de script retorna o carimbo de data/hora do cabeçalho XML de um arquivo QVD, se houver, do contrário retorna NULL. No carimbo de data/hora, a hora é fornecida em UTC.

```
QvdCreateTime (filename)
```

QvdFieldName
Esta função de script retorna o nome do número do campo **fieldno** em um arquivo QVD. Se o campo não existir, será retornado NULL.

```
QvdFieldName (filename , fieldno)
```

QvdNoOfFields
Essa função de script retorna o número de campos em um arquivo QVD.

```
QvdNoOfFields (filename)
```

QvdNoOfRecords

Essa função de script retorna o número de registros atualmente presentes em um arquivo QVD.

```
QvdNoOfRecords (filename)
```

QvdTableName

Essa função de script retorna o nome da tabela armazenada em um arquivo QVD.

```
QvdTableName (filename)
```

Attribute

Essa função de script retorna o valor de meta tags de diferentes arquivos de mídia como texto. Há suporte para os seguintes formatos de arquivo: MP3, WMA, WMV, PNG e JPG. Se o arquivo **filename** não existir, não for um formato de arquivo suportado ou não contiver uma meta tag chamada **attributename**, NULL será retornado.

Sintaxe:

```
Attribute(filename, attributename)
```

Um grande número de meta tags pode ser lido. Os exemplos neste tópico mostram quais tags podem ser lidas para os respectivos tipos de arquivos suportados.

*Você só pode ler meta tags salvas no arquivo de acordo com a especificação relevante, por exemplo, ID2v3 para arquivos MP3 ou EXIF para arquivos JPG, não a meta informação salva no **Windows File Explorer**.*

Argumentos:

Argumentos

Argumento	Descrição
filename	O nome de um arquivo de mídia, incluindo o caminho, se necessário, como uma conexão de dados da pasta. **Exemplo: *'lib://Table Files/'*** No modo legado de uso de scripts, os seguintes formatos de caminho também são suportados: • absoluto **Exemplo: *c:\data*** • relativo ao diretório operacional do Qlik Sense Qlik Cloud. **Exemplo: *data***
attributename	O nome de uma meta tag.

Os exemplos usam a função **GetFolderPath** para encontrar os caminhos para os arquivos. Como **GetFolderPath** apenas tem suporte no modo legado, você precisa substituir as referências a **GetFolderPath** por um caminho de conexão de dados lib:// quando você usa essa função no modo padrão ou no Qlik Sense SaaS.

Restrição de acesso do sistema de arquivo (page 1500)

Example 1: Arquivos MP3

Este script lê todas as meta tags MP3 possíveis na pasta *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

Este script lê todas as meta tags EXIF possíveis dos arquivos JPG na pasta *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: Arquivos de mídia Windows

Este script lê todas as meta tags WMA/WMV ASF possíveis na pasta *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

Este script lê todas as meta tags PNG possíveis na pasta *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

A função **ConnectString()** retorna o nome da conexão de dados ativa para conexões ODBC ou OLE DB . A função retorna uma string vazia se nenhum comando **connect** tiver sido executado ou após uma declaração **disconnect**.

Sintaxe:

```
ConnectString()
```

Exemplos e resultados:

Exemplos de scripts

Exemplo	Resultado

`LIB CONNECT TO 'Tutorial ODBC';`
 	
`ConnectString:`
 	
`Load ConnectString() as ConnectString AutoGenerate 1;`
	Retorna "Tutorial ODBC" no campo ConnectString.
	
	Esse exemplo assume que você tenha uma conexão de dados disponíveis chamada Tutorial ODBC.

FileBaseName

A função **FileBaseName** retorna uma string contendo o nome do arquivo de tabela que está sendo lido no momento, sem caminho ou extensão.

Sintaxe:

```
FileBaseName()
```

Exemplos e resultados:

Exemplos de scripts

Exemplo	Resultado

`LOAD *, filebasename( ) as X from`
 	
`C:\UserFiles\abc.txt`
	Retornará “abc” no campo X em cada leitura de registro.

FileDir

A função **FileDir** retorna uma string contendo o caminho do diretório do arquivo de tabela que está sendo lido no momento.

Sintaxe:

```
FileDir()
```

Esta função suporta apenas as conexões de dados da pasta no modo padrão.

Exemplos e resultados:

Exemplos de scripts

Exemplo	Resultado
`Load *, filedir( ) as X from C:\UserFiles\abc.txt`	Retornará “C:\UserFiles” no campo X em cada leitura de registro.

FileExtension

A função **FileExtension** retorna uma string contendo a extensão do arquivo de tabela que está sendo lido no momento.

Sintaxe:

```
FileExtension()
```

Exemplos e resultados:

Exemplos de scripts

Exemplo	Resultado
`LOAD *, FileExtension( ) as X from C:\UserFiles\abc.txt`	Retornará “txt” no campo X em cada leitura de registro.

FileName

A função **FileName** retorna uma string contendo o nome do arquivo de tabela que está sendo lido no momento, sem caminho, mas incluindo a extensão.

Sintaxe:

```
FileName()
```

Exemplos e resultados:

Exemplos de scripts

Exemplo	Resultado

`LOAD *, FileName( ) as X from`
 	
`C:\UserFiles\abc.txt`
	Retornará “ 'abc.txt'” no campo X em cada leitura de registro.

FilePath

A função **FilePath** retorna uma string contendo o caminho completo do arquivo de tabela que está sendo lido no momento.

Sintaxe:

```
FilePath()
```

Esta função suporta apenas as conexões de dados da pasta no modo padrão.

Exemplos e resultados:

Exemplos de scripts

Exemplo	Resultado

`Load *, FilePath( ) as X from`
 	
`C:\UserFiles\abc.txt`
	Retornará “ 'C:\UserFiles\abc.txt'” no campo X em cada leitura de registro.

FileSize

A função **FileSize** retorna um inteiro contendo o tamanho em bytes do arquivo filename ou, se nenhum filename for especificado, retorna o do arquivo de tabela que está sendo lido no momento.

Sintaxe:

```
FileSize([filename])
```

Argumentos:

Argumentos

Argumento	Descrição
filename	O nome de um arquivo, se necessário incluindo o caminho, como uma pasta ou conexão de dados de arquivo da web. Se você não especificar um nome de arquivo, o arquivo de tabela que está sendo lido no momento será usado. **Exemplo: *'lib://Table Files/'*** No modo legado de uso de scripts, os seguintes formatos de caminho também são suportados: • absoluto **Exemplo: *c:\data*** • relativo ao diretório operacional do Qlik Sense Qlik Cloud. **Exemplo: *data*** • Endereço de URL (HTTP ou FTP) que aponte para uma localização na Internet ou em uma intranet. **Exemplo: *http://www.qlik.com***

Exemplos e resultados:

Exemplos de scripts

Exemplo	Resultado
`LOAD *, FileSize( ) as X from abc.txt;`	Retornará o tamanho do arquivo especificado (abc.txt) como um inteiro no campo X em cada leitura de registro.
`FileSize( 'lib://DataFiles/xyz.xls' )`	Retornará o tamanho do arquivo xyz.xls.

FileTime

A função **FileTime** retorna um carimbo de data/hora no formato UTC da última modificação de um arquivo especificado. Se um arquivo não for especificado, a função retornará um carimbo de data/hora em UTC da última modificação do arquivo de tabela atualmente lido.

Sintaxe:

```
FileTime([ filename ])
```

Argumentos:

Argumentos

Argumento	Descrição
filename	O nome de um arquivo, incluindo o caminho, se necessário, como uma pasta ou uma conexão de dados do arquivo da web. **Exemplo: *'lib://Table Files/'*** No modo legado de uso de scripts, os seguintes formatos de caminho também são suportados: • absoluto **Exemplo: *c:\data*** • relativo ao diretório operacional do Qlik Sense Qlik Cloud. **Exemplo: *data*** • Endereço de URL (HTTP ou FTP) que aponte para uma localização na Internet ou em uma intranet. **Exemplo: *http://www.qlik.com***

Exemplos e resultados:

Exemplos de script

Exemplo	Resultado
`LOAD *, FileTime( ) as X from abc.txt;`	Retornará a data e hora da última modificação do arquivo (abc.txt) no campo X em cada leitura de registro.
`FileTime( 'xyz.xls' )`	Retornará a data/hora da última modificação do arquivo xyz.xls.

GetFolderPath

A função **GetFolderPath** retorna o valor da função do Microsoft Windows *SHGetFolderPath*. Esta função admite como entrada o nome de uma pasta doMicrosoft Windows e retorna o caminho completo da pasta.

Esta função não é suportada no modo padrão. .

Sintaxe:

```
GetFolderPath(foldername)
```

Argumentos:

Argumentos

Argumento	Descrição
foldername	Nome da pasta do Microsoft Windows. O nome da pasta não deve conter espaço. Qualquer espaço no nome da pasta visto no Windows Explorer deve ser removido do nome da pasta. Exemplos: *MyMusic* *MyDocuments*

Exemplos e resultados:

O objetivo deste exemplo é obter os caminhos das seguintes pastas do Microsoft Windows: *MyMusic*, *MyPictures* e *Windows*. Adicione o script de exemplo ao seu aplicativo e execute-o.

```
LOAD
 GetFolderPath('MyMusic') as MyMusic,
 GetFolderPath('MyPictures') as MyPictures,
 GetFolderPath('Windows') as Windows
AutoGenerate 1;
```

Quando o aplicativo recarregar, os campos *MyMusic*, *MyPictures* e *Windows* serão adicionados ao modelo de dados. Cada campo contém o caminho para a pasta definida na entrada. Por exemplo:

- *C:\Users\smu\Music* for the folder *MyMusic*
- *C:\Users\smu\Pictures* for the folder *MyPictures*
- *C:\Windows* for the folder *Windows*

QvdCreateTime

Esta função de script retorna o carimbo de data/hora do cabeçalho XML de um arquivo QVD, se houver, do contrário retorna NULL. No carimbo de data/hora, a hora é fornecida em UTC.

Sintaxe:

```
QvdCreateTime(filename)
```

Argumentos:

Argumentos

Argumento	Descrição
filename	O nome de um arquivo QVD, se necessário, incluindo o caminho, como uma pasta ou conexão de dados da web. **Exemplo: *'lib://Table Files/'*** No modo legado de uso de scripts, os seguintes formatos de caminho também são suportados: • absoluto **Exemplo: *c:\data*** • relativo ao diretório operacional do Qlik Sense Qlik Cloud. **Exemplo: *data*** • Endereço de URL (HTTP ou FTP) que aponte para uma localização na Internet ou em uma intranet. **Exemplo: *http://www.qlik.com***

Exemplo:

```
QvdCreateTime('MyFile.qvd')

QvdCreateTime('C:\MyDir\MyFile.qvd')

QvdCreateTime('lib://DataFiles/MyFile.qvd')
```

QvdFieldName

Esta função de script retorna o nome do número do campo **fieldno** em um arquivo QVD. Se o campo não existir, será retornado NULL.

Sintaxe:

```
QvdFieldName(filename , fieldno)
```

Argumentos:

Argumentos

Argumento	Descrição
filename	O nome de um arquivo QVD, se necessário, incluindo o caminho, como uma pasta ou conexão de dados da web. **Exemplo: *'lib://Table Files/'*** No modo legado de uso de scripts, os seguintes formatos de caminho também são suportados: • absoluto **Exemplo: *c:\data*** • relativo ao diretório operacional do Qlik Sense Qlik Cloud. **Exemplo: *data*** • Endereço de URL (HTTP ou FTP) que aponte para uma localização na Internet ou em uma intranet. **Exemplo: *http://www.qlik.com***
fieldno	O número do campo na tabela contida no arquivo QVD.

Exemplos:

```
QvdFieldName ('MyFile.qvd', 5)

QvdFieldName ('C:\MyDir\MyFile.qvd', 5)

QvdFieldName ('lib://DataFiles/MyFile.qvd', 5)
```

Todos os três exemplos retornam o nome do quinto campo da tabela contida no arquivo QVD.

QvdNoOfFields

Essa função de script retorna o número de campos em um arquivo QVD.

Sintaxe:

```
QvdNoOfFields(filename)
```

Argumentos:

Argumentos

Argumento	Descrição
filename	O nome de um arquivo QVD, se necessário, incluindo o caminho, como uma pasta ou conexão de dados da web. **Exemplo: *'lib://Table Files/'*** No modo legado de uso de scripts, os seguintes formatos de caminho também são suportados: • absoluto **Exemplo: *c:\data*** • relativo ao diretório operacional do Qlik Sense Qlik Cloud. **Exemplo: *data*** • Endereço de URL (HTTP ou FTP) que aponte para uma localização na Internet ou em uma intranet. **Exemplo: *http://www.qlik.com***

Exemplos:

```
QvdNoOfFields ('MyFile.qvd')

QvdNoOfFields ('C:\MyDir\MyFile.qvd')

QvdNoOfFields ('lib://DataFiles/MyFile.qvd')
```

QvdNoOfRecords

Exemplo: Essa função de script retorna o número de registros atualmente presentes em um arquivo QVD.

Sintaxe:

```
QvdNoOfRecords(filename)
```

Argumentos:

Argumentos

Argumento	Descrição
filename	O nome de um arquivo QVD, se necessário, incluindo o caminho, como uma pasta ou conexão de dados da web. **Exemplo: *'lib://Table Files/'*** No modo legado de uso de scripts, os seguintes formatos de caminho também são suportados: • absoluto **Exemplo: *c:\data*** • relativo ao diretório operacional do Qlik Sense Qlik Cloud. **Exemplo: *data*** • Endereço de URL (HTTP ou FTP) que aponte para uma localização na Internet ou em uma intranet. **Exemplo: *http://www.qlik.com***

Exemplos:

```
QvdNoOfRecords ('MyFile.qvd')

QvdNoOfRecords ('C:\MyDir\MyFile.qvd')

QvdNoOfRecords ('lib://DataFiles/MyFile.qvd')
```

QvdTableName

Essa função de script retorna o nome da tabela armazenada em um arquivo QVD.

Sintaxe:

```
QvdTableName(filename)
```

Argumentos:

Argumentos

Argumento	Descrição
filename	O nome de um arquivo QVD, se necessário, incluindo o caminho, como uma pasta ou conexão de dados da web. **Exemplo: *'lib://Table Files/'*** No modo legado de uso de scripts, os seguintes formatos de caminho também são suportados: • absoluto **Exemplo: *c:\data*** • relativo ao diretório operacional do Qlik Sense Qlik Cloud. **Exemplo: *data*** • Endereço de URL (HTTP ou FTP) que aponte para uma localização na Internet ou em uma intranet. **Exemplo: *http://www.qlik.com***

Exemplos:

```
QvdTableName ('MyFile.qvd')

QvdTableName ('C:\MyDir\MyFile.qvd')

QvdTableName ('lib://data\MyFile.qvd')
```

5.11 Funções financeiras

As funções financeiras podem ser usadas no script de carregamento de dados e em expressões do gráfico para calcular os pagamentos e taxas de juros.
Para todos os argumentos, o dinheiro pago será representado por números negativos. O dinheiro recebido será representado por números positivos.
Estão listados aqui os argumentos usados nas funções financeiras (exceto os que começam com **range**-).

*Para todas as funções financeiras, é imprescindível ser consistente ao especificar unidades para **rate** e **nper**. Se você fizer pagamentos mensais de um empréstimo de cinco anos com juros de 6% ao ano, use 0,005 (6%/12) para **rate** e 60 (5*12) para **nper**. Se você fizer pagamentos anuais do mesmo empréstimo, use 6% para **rate** e 5 para **nper**.*

Visão geral das funções financeiras

Cada função é descrita adicionalmente após a visão geral. Você também pode clicar no nome da função na sintaxe para acessar imediatamente os detalhes dessa função específica.

FV
Esta função retorna o valor futuro de um investimento com base em pagamentos constantes periódicos e uma taxa de juros anual.

```
FV (rate, nper, pmt [ ,pv [ , type ] ])
```

nPer
Esta função retorna o número de períodos de um investimento com base em pagamentos constantes periódicos e em uma taxa de juros constante.

```
nPer (rate, pmt, pv [ ,fv [ , type ] ])
```

Pmt
Esta função retorna o pagamento de um empréstimo com base em pagamentos constantes periódicos e em uma taxa de juros constante. Não pode ser alterado no decorrer da anuidade. Um pagamento é indicado como um número negativo, por exemplo, -20.

```
Pmt (rate, nper, pv [ ,fv [ , type ] ] )
```

PV
Esta retorna o valor atual de um investimento.

```
PV (rate, nper, pmt [ ,fv [ , type ] ])
```

Rate
Esta função retorna a taxa de juros por período na anuidade. O resultado apresenta um formato numérico padrão de dois decimais **Fix** e %.

```
Rate (nper, pmt , pv [ ,fv [ , type ] ])
```

BlackAndSchole

O modelo Black and Scholes é um modelo matemático de instrumentos derivativos no mercado financeiro. A fórmula calcula o valor teórico de uma opção. No Qlik Sense, a função **BlackAndSchole** retorna o valor de acordo com a fórmula não modificada de Black and Scholes (opções de estilo europeu).

```
BlackAndSchole(strike , time_left , underlying_price , vol , risk_free_rate , type)
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
strike	O preço de compra futuro da ação.
time_left	O número de períodos de tempo restantes.
underlying_price	O valor atual da ação.
vol	Volatilidade (do preço da ação) expressa em porcentagem na forma decimal, por período de tempo.
risk_free_rate	A taxa livre de risco expressa em porcentagem na forma decimal, por período de tempo.
call_or_put	O tipo de opção: 'c', 'call' ou qualquer valor numérico diferente de zero para opções de call 'p', 'put' ou 0 para opções de put

Limitações:

O valor de strike, time_left e underlying_price deve ser >0.

O valor de vol e risk_free_rate deve ser: <0 ou >0.

Exemplos e resultados:

Exemplos de scripts

Exemplo	Resultado
`BlackAndScholes(130, 4, 68.5, 0.4, 0.04, 'call')` Calcula o preço teórico de uma opção de comprar uma ação valendo 68,5 hoje, a um valor de 130 em 4 anos. A fórmula usa uma volatilidade de 0,4 (40%) ao ano e uma taxa de juros livre de risco de 0,04 (4%).	Retorna 11,245

FV

Esta função retorna o valor futuro de um investimento com base em pagamentos constantes periódicos e uma taxa de juros anual.

Sintaxe:

```
FV(rate, nper, pmt [ ,pv [ , type ] ])
```

Tipo de dados de retorno: numérico. Por padrão, o resultado será formatado como moeda..

Argumentos:

Argumentos

Argumento	Descrição
rate	A taxa de juros por período.
nper	O número total de períodos de pagamento de uma anuidade.
pmt	O pagamento feito em cada período. Não pode ser alterado no decorrer da anuidade. Um pagamento é indicado como um número negativo, por exemplo, -20.
pv	O valor presente ou a quantia total atual correspondente a uma série de pagamentos futuros. Se **pv** for omitido, o valor 0 (zero) será assumido.
type	Deverá ser 0 se os vencimentos dos pagamentos estiverem programados para o fim do período; deverá ser 1 se os vencimentos estiverem programados para o início. Se **type** for omitido, o valor 0 será assumido.

Exemplos e resultados:

Exemplo de scripts

Exemplo	Resultado
Você está pagando um novo eletrodoméstico em 36 prestações mensais de US$ 20. A taxa de juros é de 6% ao ano. A fatura chega no fim de cada mês. Qual será o total investido após o pagamento da última fatura? `FV(0.005,36,-20)`	Retorna $786.72

nPer

Esta função retorna o número de períodos de um investimento com base em pagamentos constantes periódicos e em uma taxa de juros constante.

Sintaxe:

```
nPer(rate, pmt, pv [ ,fv [ , type ] ])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
rate	A taxa de juros por período.
nper	O número total de períodos de pagamento de uma anuidade.

Argumento	Descrição
pmt	O pagamento feito em cada período. Não pode ser alterado no decorrer da anuidade. Um pagamento é indicado como um número negativo, por exemplo, -20.
pv	O valor presente ou a quantia total atual correspondente a uma série de pagamentos futuros. Se **pv** for omitido, o valor 0 (zero) será assumido.
fv	O valor futuro ou um saldo em dinheiro que se deseja obter após o último pagamento ter sido efetuado. Se **fv** for omitido, o valor 0 será assumido.
type	Deverá ser 0 se os vencimentos dos pagamentos estiverem programados para o fim do período; deverá ser 1 se os vencimentos estiverem programados para o início. Se **type** for omitido, o valor 0 será assumido.

Exemplos e resultados:

Exemplo de scripts

Exemplo	Resultado
Você quer vender um eletrodoméstico em prestações mensais de US$ 20. A taxa de juros é de 6% ao ano. A fatura chega no fim de cada mês. Quantos períodos serão necessários se o valor da soma recebida após o pagamento da última fatura tiver que ser igual a US$ 800? `nPer(0.005,-20,0,800)`	Retorna 36,56

Pmt

Esta função retorna o pagamento de um empréstimo com base em pagamentos constantes periódicos e em uma taxa de juros constante. Não pode ser alterado no decorrer da anuidade. Um pagamento é indicado como um número negativo, por exemplo, -20.

```
Pmt(rate, nper, pv [ ,fv [ , type ] ] )
```

Tipo de dados de retorno: numérico. Por padrão, o resultado será formatado como moeda..

Para obter o total pago durante o empréstimo, multiplique o valor retornado de **pmt** por **nper**.

Argumentos:

Argumentos

Argumento	Descrição
rate	A taxa de juros por período.
nper	O número total de períodos de pagamento de uma anuidade.
pv	O valor presente ou a quantia total atual correspondente a uma série de pagamentos futuros. Se **pv** for omitido, o valor 0 (zero) será assumido.

Argumento	Descrição
fv	O valor futuro ou um saldo em dinheiro que se deseja obter após o último pagamento ter sido efetuado. Se **fv** for omitido, o valor 0 será assumido.
type	Deverá ser 0 se os vencimentos dos pagamentos estiverem programados para o fim do período; deverá ser 1 se os vencimentos estiverem programados para o início. Se **type** for omitido, o valor 0 será assumido.

Exemplos e resultados:

Exemplos de scripts

Exemplo	Resultado
A fórmula a seguir retorna o pagamento mensal de um empréstimo de US$ 20.000, a uma taxa de 10% ao ano, a ser pago em 8 meses: `Pmt(0.1/12,8,20000)`	Retorna -$2,594.66
Para o mesmo empréstimo, se o vencimento do pagamento fosse no início do período, o pagamento seria: `Pmt(0.1/12,8,20000,0,1)`	Retorna -$2,573.21

PV

Esta retorna o valor atual de um investimento.

```
PV(rate, nper, pmt [ ,fv [ , type ] ])
```

Tipo de dados de retorno: numérico. Por padrão, o resultado será formatado como moeda..

O valor presente é a quantia total correspondente a uma série de pagamentos futuros. Por exemplo, quando você assume um empréstimo, o valor do empréstimo é o valor atual devido a quem empresta.

Argumentos:

Argumentos

Argumento	Descrição
rate	A taxa de juros por período.
nper	O número total de períodos de pagamento de uma anuidade.
pmt	O pagamento feito em cada período. Não pode ser alterado no decorrer da anuidade. Um pagamento é indicado como um número negativo, por exemplo, -20.
fv	O valor futuro ou um saldo em dinheiro que se deseja obter após o último pagamento ter sido efetuado. Se **fv** for omitido, o valor 0 será assumido.
type	Deverá ser 0 se os vencimentos dos pagamentos estiverem programados para o fim do período; deverá ser 1 se os vencimentos estiverem programados para o início. Se **type** for omitido, o valor 0 será assumido.

Exemplos e resultados:

Exemplo de scripts

Exemplo	Resultado
Qual é o valor presente de uma dívida, quando você deve pagar US$100 no final de cada mês durante um período de cinco anos, dado uma taxa de juros de 7%? `PV(0.07/12,12*5,-100,0,0)`	Retorna $5,050.20

Rate

Esta função retorna a taxa de juros por período na anuidade. O resultado apresenta um formato numérico padrão de dois decimais **Fix** e %.

Sintaxe:

```
Rate(nper, pmt , pv [ ,fv [ , type ] ])
```

Tipo de dados de retorno: numérico.

A **rate** é calculada por iteração e pode ter zero ou mais soluções. Se os resultados sucessivos de **rate** não convergirem, será retornado um valor NULL.

Argumentos:

Argumentos

Argumento	Descrição
nper	O número total de períodos de pagamento de uma anuidade.
pmt	O pagamento feito em cada período. Não pode ser alterado no decorrer da anuidade. Um pagamento é indicado como um número negativo, por exemplo, -20.
pv	O valor presente ou a quantia total atual correspondente a uma série de pagamentos futuros. Se **pv** for omitido, o valor 0 (zero) será assumido.
fv	O valor futuro ou um saldo em dinheiro que se deseja obter após o último pagamento ter sido efetuado. Se **fv** for omitido, o valor 0 será assumido.
type	Deverá ser 0 se os vencimentos dos pagamentos estiverem programados para o fim do período; deverá ser 1 se os vencimentos estiverem programados para o início. Se **type** for omitido, o valor 0 será assumido.

Exemplos e resultados:

Exemplo de scripts

Exemplo	Resultado
Qual é a taxa de juros de um empréstimo de cinco anos, com anuidade equivalente a US$ 10.000 e pagamentos mensais de US$ 300? `Rate(60,-300,10000)`	Retorna 2.00%

5.12 Funções de formato

As funções de formato impõem o formato de exibição nas expressões ou campos numéricos de entrada. Dependendo do tipo de dados, você poderá especificar os caracteres para o separador decimal, de milhar e outros.

As funções retornam um valor duplo com os valores numéricos e da string, mas podem ser consideradas como a realização de uma conversão de número para string. **Dual()** é um caso especial, mas as outras funções de formato pegam o valor numérico da expressão de entrada e geram uma string representando o número.

Em contraste, as funções de interpretação fazem o oposto: pegam as expressões da string e as avaliam como números, especificando o formato do número resultante.

As funções podem ser usadas em scripts de carga de dados ou em expressões de gráfico.

Todas as representações numéricas são dadas com um ponto decimal como separador decimal.

Visão geral das funções de formatação

Cada função é descrita adicionalmente após a visão geral. Você também pode clicar no nome da função na sintaxe para acessar imediatamente os detalhes dessa função específica.

ApplyCodepage
ApplyCodepage() aplica um conjunto de caracteres de página de código diferente ao campo ou texto estabelecido na expressão. O argumento **codepage** deve estar no formato numérico.

```
ApplyCodepage (text, codepage)
```

Date
Date() formata uma expressão como uma data usando o formato definido nas variáveis de sistema no script de carga de dados ou sistema operacional, ou uma string de formato, se fornecida.

```
Date (number[, format])
```

Dual
Dual() combina um número e uma string em um único registro, de modo que representação numérica do registro possa ser usada para classificar e calcular propósitos, enquanto o valor da string pode ser usado para exibir propósitos.

```
Dual (text, number)
```

Interval
Interval() formata um número como um intervalo de tempo, usando o formato nas variáveis de sistema no script de carga de dados ou o sistema operacional, ou uma string de formato, se fornecida.

```
Interval (number[, format])
```

Money

Money() formata uma expressão numericamente como um valor de moeda no formato numérico definido nas variáveis de sistema, definido no script de carga dos dados ou no sistema operacional, a não ser que uma string de formato seja fornecida, bem como separadores decimais e de milhar opcionais.

```
Money (number[, format[, dec_sep [, thou_sep]]])
```

Num

Num() formata um número, ou seja, converte o valor numérico da entrada para exibir texto usando o formato especificado no segundo parâmetro. Se o segundo parâmetro for omitido, ele usará os separadores decimais e de milhar definidos no script de carregamento de dados. Símbolos de separadores decimais e de milhar personalizados são parâmetros opcionais.

```
Num (number[, format[, dec_sep [, thou_sep]]])
```

Time

Time() formata uma expressão como um valor de tempo, no formato de tempo definido nas variáveis de sistema no script de carga de dados ou no sistema operacional, a não ser que uma string de formato seja fornecida.

```
Time (number[, format])
```

Timestamp

TimeStamp() formata uma expressão como um valor de data e hora, no formato de carimbo de data e hora definido nas variáveis de sistema no script de carga de dados ou no sistema operacional, a não ser que uma string de formato seja fornecida.

```
Timestamp (number[, format])
```

Consulte também:

Funções de interpretação (page 1270)

ApplyCodepage

ApplyCodepage() aplica um conjunto de caracteres de página de código diferente ao campo ou texto estabelecido na expressão. O argumento **codepage** deve estar no formato numérico.

Embora ApplyCodepage possa ser usado em expressões de gráfico, ele é mais comumente usado como uma função de script no Editor de carregamento de dados. Por exemplo, conforme você carrega os arquivos que podem ter sido salvos em conjuntos de caracteres diferentes fora de controle, você pode aplicar a página de código que representa o conjunto de caracteres que você precisa.

Sintaxe:

```
ApplyCodepage(text, codepage)
```

Tipo de dados de retorno: caractere

Argumentos:

Argumentos

Argumento	Descrição
text	Campo ou texto ao qual você deseja aplicar uma página de código diferente, fornecida pelo argumento **codepage**.
codepage	Número que representa a página de código a ser aplicada ao campo ou expressão fornecida por **text**.

Exemplos e resultados:

Exemplos de scripts

Exemplo	Resultado
`LOAD ApplyCodepage (ROWX,1253) as GreekProduct, ApplyCodepage (ROWY, 1255) as HebrewProduct, ApplyCodepage (ROWZ, 65001) as EnglishProduct; SQL SELECT ROWX, ROWY, ROWZ From Products;`	Ao carregar a partir do SQL, a fonte pode ter uma mistura de conjuntos de caracteres diferentes: Cyrillic, Hebrew e assim por diante, a partir do formato UTF-8. Os mesmos precisariam ser carregados linha por linha, aplicando-se uma página de código diferente para cada linha. O valor **codepage** 1.253 representa o conjunto de caracteres do Windows Greek, o valor 1.255 representa Hebrew e o valor 65.001 representa caracteres latim padrão UTF-8.

Consulte também: *Conjunto de caracteres (page 168)*

Date

Date() formata uma expressão como uma data usando o formato definido nas variáveis de sistema no script de carga de dados ou sistema operacional, ou uma string de formato, se fornecida.

Sintaxe:

```
Date(number[, format])
```

Tipo de dados de retorno: dual

Argumentos:

Argumentos

Argumento	Descrição
number	O número a ser formatado.
format	String descrevendo o formato da string resultante. Se nenhuma string de formato for fornecida, será usado o formato de data definido nas variáveis do sistema no script de carregamento de dados ou no sistema operacional.

Exemplos e resultados:
Os exemplos abaixo supõem as seguintes configurações padrão:

- Configuração de data 1: YY-MM-DD
- Configuração de data 2: M/D/YY

Exemplo:

```
Date( A )
```
em que A=35648

Tabela de resultados

Resultados	Configuração 1	Configuração 2
String:	97-08-06	8/6/97
Número:	35648	35648

Exemplo:

```
Date( A, 'YY.MM.DD' )
```
em que A=35648

Tabela de resultados

Resultados	Configuração 1	Configuração 2
String:	97.08.06	97.08.06
Número:	35648	35648

Exemplo:

```
Date( A, 'DD.MM.YYYY' )
```
em que A=35648.375

Tabela de resultados

Resultados	Configuração 1	Configuração 2
String:	06.08.1997	06.08.1997
Número:	35648.375	35648.375

Exemplo:

```
Date( A, 'YY.MM.DD' )
```
em que A=8/6/97

Tabela de resultados

Resultados	Configuração 1	Configuração 2
String:	NULL (nada)	97.08.06
Número:	NULL	35648

Dual

Dual() combina um número e uma string em um único registro, de modo que representação numérica do registro possa ser usada para classificar e calcular propósitos, enquanto o valor da string pode ser usado para exibir propósitos.

Sintaxe:

```
Dual(text, number)
```

Tipo de dados de retorno: dual

Argumentos:

Argumentos

Argumento	Descrição
text	O valor da string a ser usado em combinação com o argumento numérico.
number	O número a ser usado em combinação com a string no argumento numérico.

No Qlik Sense, todos os valores de campo são potencialmente valores duais. Isso significa que os valores de campo podem ter tanto um valor numérico quanto um valor textual. Um exemplo é uma data que poderia ter um valor numérico de 40908 e a representação textual '2011-12-31'.

Quando vários itens de dados lidos em um campo tiverem diferentes representações de string, mas a mesma representação numérica válida, eles compartilharão a primeira representação de string encontrada.

*Em geral, a função **dual** é usada primeiro no script, antes da leitura de outros dados no campo em questão, a fim de criar essa primeira representação de string, que será mostrada nos painéis de filtro.*

Exemplos e resultados:

Exemplos de scripts

Exemplo	**Descrição**

Adicione os seguintes exemplos ao seu script e execute-o.
`Load dual ( NameDay,NumDay ) as DayOfWeek inline`
`[ NameDay,NumDay`
`Monday,0`
`Tuesday,1`
`Wednesday,2`
`Thursday,3`
`Friday,4`
`Saturday,5`
`Sunday,6 ];`
	O campo DayOfWeek pode ser usado em uma visualização, como uma dimensão, por exemplo. Em uma tabela com os dias da semana, eles são classificados automaticamente na sequência numérica correta, em vez da ordem alfabética.

`Load Dual('Q' & Ceil (Month(Now())/3), Ceil (Month(Now())/3)) as Quarter AutoGenerate 1;`	Este exemplo localiza o trimestre atual. Ele é exibido como Q1 quando a função **Now()** é executada nos primeiros três meses do ano, Q2 para o segundo trimestre, e assim por diante. No entanto, quando usado em classificação, o campo Quarter se comportará como valor numérico: 1 a 4.
`Dual('Q' & Ceil(Month (Date)/3), Ceil(Month (Date)/3)) as Quarter`	Como no exemplo anterior, o campo Quarter é criado com os valores de texto 'Q1' a 'Q4' e são atribuídos os valores numéricos 1 a 4. Para usar isso no script, os valores de Date devem ser carregados.
`Dual(WeekYear(Date) & '-W' & Week(Date), WeekStart(Date)) as YearWeek`	Esse exemplo criará um campo YearWeek com os valores textuais com o formato '2012-W22' e, ao mesmo tempo, atribuirá um valor numérico correspondente ao número de data do primeiro dia da semana, por exemplo: 41057. Para usar isso no script, os valores de Date devem ser carregados.

Interval

Interval() formata um número como um intervalo de tempo, usando o formato nas variáveis de sistema no script de carga de dados ou o sistema operacional, ou uma string de formato, se fornecida.

Os intervalos podem ser formatados como hora, dia ou como uma combinação de dias, horas, minutos, segundos e frações de segundos.

Sintaxe:

```
Interval(number[, format])
```

Tipo de dados de retorno: dual

Argumentos:

Argumentos

Argumento	Descrição
number	O número a ser formatado.
format	String descrevendo como a string resultante do intervalo será formatada. Se omitida, serão utilizados os formatos de data abreviada, formato de hora e separador decimal configurados no sistema operacional.

Exemplos e resultados:
Os exemplos abaixo supõem as seguintes configurações padrão:

- Configuração do formato de data 1: YY-MM-DD
- Configuração do formato de data 2: hh:mm:ss
- Separador de número decimal: .

Tabela de resultados

Exemplo	String	Número
`Interval( A )` em que A=0,375	09:00:00	0.375
`Interval( A )` em que A=1.375	33:00:00	1.375
`Interval( A, 'D hh:mm' )` em que A=1.375	1 09:00	1.375
`Interval( A-B, 'D hh:mm' )` em que A=97-08-06 09:00:00 e B=96-08-06 00:00:00	365 09:00	365.375

Money

Money() formata uma expressão numericamente como um valor de moeda no formato numérico definido nas variáveis de sistema, definido no script de carga dos dados ou no sistema operacional, a não ser que uma string de formato seja fornecida, bem como separadores decimais e de milhar opcionais.

Sintaxe:

```
Money(number[, format[, dec_sep[, thou_sep]]])
```

Tipo de dados de retorno: dual

Argumentos:

Argumentos

Argumento	Descrição
number	O número a ser formatado.
format	String descrevendo como a string de moeda resultante será formatada.
dec_sep	String especificando o separador de número decimal.
thou_sep	String especificando o separador de número milhar.

Se os parâmetros de 2 a 4 forem omitidos, será utilizado o formato de moeda definido no sistema operacional.

Exemplos e resultados:
Os exemplos abaixo supõem as seguintes configurações padrão:

- Configuração de MoneyFormat 1: kr ##0,00, MoneyThousandSep' '
- Configuração de MoneyFormat 2: $ #,##0.00, MoneyThousandSep','

Exemplo:

```
Money( A )
```
em que A=35648

Tabela de resultados

Resultados	Configuração 1	Configuração 2
String:	kr 35 648,00	$ 35,648.00
Número:	35648.00	35648.00

Exemplo:

```
Money( A, '#,##0 ¥', '.' , ',' )
```
em que A=3564800

Tabela de resultados

Resultados	Configuração 1	Configuração 2
String:	3,564,800 ¥	3,564,800 ¥
Número:	3564800	3564800

Num

Num() formata um número, ou seja, converte o valor numérico da entrada para exibir texto usando o formato especificado no segundo parâmetro. Se o segundo parâmetro for omitido, ele usará os separadores decimais e de milhar definidos no script de carregamento de dados. Símbolos de separadores decimais e de milhar personalizados são parâmetros opcionais.

Sintaxe:

```
Num(number[, format[, dec_sep [, thou_sep]]])
```

Tipo de dados de retorno: dual

A função Num retorna um valor duplo como a string e o valor numérico. Ela usa o valor numérico da expressão de entrada e gera uma string representando o número.

Argumentos:

Argumentos

Argumento	Descrição
number	O número a ser formatado.
format	String que especifica como a string resultante deve ser formatada. Se omitida, os separadores decimais e de milhar definidos no script de carregamento de dados serão usados.
dec_sep	String que especifica o separador de número decimal. Se omitida, o valor da variável DecimalSep definido no script de carregamento de dados será usado.
thou_sep	String que especifica o separador de número milhar. Se omitida, o valor da variável ThousandSep que é definido no script de carregamento de dados será usado.

Exemplo: Expressão de gráfico

Exemplo:

A tabela a seguir mostra os resultados quando o campo A é igual a 35648.312.

Resultados

A	Resultado
Num(A)	35648.312 (depende de variáveis de ambiente no script)
Num(A, '0.0', '.')	35648.3
Num(A, '0,00', ',')	35648,31
Num(A, '#,##0.0', '.', ',')	35,648.3
Num(A, '# ##0', ',', ' ')	35 648

Exemplo: Script de carregamento

Script de carregamento

Num pode ser usado no script de carregamento para formatar um número, mesmo que os separadores de milhar e decimal já estejam definidos no script. O script de carregamento abaixo inclui separadores de milhar e decimal específicos, mas depois usa *Num* para formatar dados de maneiras diferentes.

No **Editor de carregamento de dados**, crie uma nova seção e, em seguida, adicione o script de exemplo e execute-o. Em seguida, adicione pelo menos os campos listados na coluna de resultados a uma pasta em seu aplicativo para ver o resultado.

```
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
];
```

Tabela Qlik Sense mostrando os resultados de diferentes usos da função *Num* no script de carregamento. A quarta coluna da tabela contém um uso de formatação incorreto, para fins de exemplo.

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

Exemplo: Script de carregamento

Script de carregamento

Num pode ser usado em um script de carregamento para formatar um número como uma porcentagem.

No **Editor de carregamento de dados**, crie uma nova seção e, em seguida, adicione o script de exemplo e execute-o. Em seguida, adicione pelo menos os campos listados na coluna de resultados a uma pasta para ver o resultado.

```
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
];
```

Tabela Qlik Sense mostrando os resultados da função *Num* sendo usada no script de carregamento para formatar porcentagens.

Discount	Discount #,##0%
0.333333333333333	33%
0.22	22%
0	0%
.14	14%
0.1	10%
0	0%
75	7,500%

Time

Time() formata uma expressão como um valor de tempo, no formato de tempo definido nas variáveis de sistema no script de carga de dados ou no sistema operacional, a não ser que uma string de formato seja fornecida.

Sintaxe:

```
Time(number[, format])
```

Tipo de dados de retorno: dual

Argumentos:

Argumentos

Argumento	Descrição
number	O número a ser formatado.
format	String descrevendo como a string de hora resultante será formatada. Se omitida, serão utilizados formatos de data abreviada, formato de hora e separador decimal configurados no sistema operacional.

Exemplos e resultados:
Os exemplos abaixo supõem as seguintes configurações padrão:

- Configuração do formato de hora 1: hh:mm:ss
- Configuração do formato de hora 2: hh.mm.ss

Exemplo:

```
Time( A )
```
em que A=0,375

Tabela de resultados

Resultados	Configuração 1	Configuração 2
String:	09:00:00	09.00.00
Número:	0.375	0.375

Exemplo:

```
Time( A )
```
em que A=35648,375

Tabela de resultados

Resultados	Configuração 1	Configuração 2
String:	09:00:00	09.00.00
Número:	35648.375	35648.375

Exemplo:

```
Time( A, 'hh-mm' )
```
em que A=0,99999

Tabela de resultados

Resultados	Configuração 1	Configuração 2
String:	23-59	23-59
Número:	0.99999	0.99999

Timestamp

TimeStamp() formata uma expressão como um valor de data e hora, no formato de carimbo de data e hora definido nas variáveis de sistema no script de carga de dados ou no sistema operacional, a não ser que uma string de formato seja fornecida.

Sintaxe:

```
Timestamp(number[, format])
```

Tipo de dados de retorno: dual

Argumentos:

Argumentos

Argumento	Descrição
number	O número a ser formatado.
format	String descrevendo como a string de carimbo de data/hora resultante será formatada. Se omitida, serão utilizados formatos de data abreviada, formato de hora e separador decimal configurados no sistema operacional.

Exemplos e resultados:
Os exemplos abaixo supõem as seguintes configurações padrão:

- Configuração de TimeStampFormat 1: YY-MM-DD hh:mm:ss
- Configuração de TimeStampFormat 2: M/D/YY hh:mm:ss

Exemplo:

```
Timestamp( A )
```
em que A=35648,375

Tabela de resultados

Resultados	Configuração 1	Configuração 2
String:	97-08-06 09:00:00	8/6/97 09:00:00
Número:	35648.375	35648.375

Exemplo:

```
Timestamp( A,'YYYY-MM-DD hh.mm')
```
em que A=35648

Tabela de resultados

Resultados	Configuração 1	Configuração 2
String:	1997-08-06 00.00	1997-08-06 00.00
Número:	35648	35648

5.13 Funções numéricas gerais

Nestas funções numéricas gerais, os argumentos são expressões em que **x** deve ser interpretado como um número de valor real. Todas as funções podem ser usadas em scripts de carga de dados ou em expressões de gráfico.

Visão geral das funções numéricas gerais

Cada função é descrita adicionalmente após a visão geral. Você também pode clicar no nome da função na sintaxe para acessar imediatamente os detalhes dessa função específica.

bitcount

BitCount() retorna quantos bits no equivalente binário de um número decimal estão definidos como 1. Ou seja, a função retorna o número de bits definidos em **integer_number**, em que **integer_number** é interpretado como um número inteiro de 32 bits com sinal.

```
BitCount(integer_number)
```

div

Div() retorna a parte inteira da divisão aritmética do primeiro argumento pelo segundo argumento. Os dois parâmetros são interpretados como números reais, isto é, não precisam ser inteiros.

```
Div (integer_number1, integer_number2)
```

fabs

Fabs() retorna o valor absoluto de **x**. O resultado é um número positivo.

```
Fabs (x)
```

fact

Fact() retorna o fatorial de um inteiro positivo **x**.

```
Fact (x)
```

frac

Frac() retorna a parte fracionária de **x**.

```
Frac (x)
```

sign

Sign() retorna 1, 0 ou -1, dependendo se **x** for um número positivo, 0 ou um número negativo.

```
Sign (x)
```

Funções de combinação e permutação

combin

Combin() retorna o número de combinações de elementos **q** que podem ser coletados de um grupo de itens **p**. Conforme representado pela fórmula: `Combin(p,q) = p! / q!(p-q)!` A ordem na qual os itens são selecionados é insignificante.

```
Combin (p, q)
```

permut

Permut() retorna o número de permutas de elementos **q** que podem ser selecionadas de um grupo de itens **p**. Conforme representado pela fórmula: `Permut(p,q) = (p)! / (p - q)!` A ordem na qual os itens são selecionados é significativa.

```
Permut (p, q)
```

Funções modulares

fmod

fmod() é uma função de módulo generalizada que retorna a parte restante da divisão de inteiro do primeiro argumento (o dividendo) pelo segundo argumento (o divisor). O resultado é um número real. Os dois argumentos são interpretados como números reais, isto é, não precisam ser inteiros.

```
Fmod (a, b)
```

mod

Mod() é uma função de módulo matemático que retorna o restante não negativo de uma divisão de inteiro. O primeiro argumento é o dividendo e o segundo é o divisor. Os dois argumentos devem ser valores inteiros.

```
Mod (integer_number1, integer_number2)
```

Funções de paridade

even

Even() retorna True (-1), se **integer_number** for um número inteiro par ou zero. Retorna False (0), se **integer_number** for um número inteiro ímpar, e NULL se **integer_number** não for um número inteiro.

```
Even (integer_number)
```

odd

Odd() retorna True (-1), se **integer_number** for um número inteiro ímpar ou zero. Retorna False (0), se **integer_number** for um número inteiro par, e NULL se **integer_number** não for um número inteiro.

```
Odd (integer_number)
```

Funções de arredondamento

ceil

Ceil() arredonda um número para o múltiplo mais próximo da **step** deslocada pelo número de **offset** .

```
Ceil (x[, step[, offset]])
```

floor

Floor() arredonda um número para baixo, para o múltiplo mais próximo da **step** deslocada pelo número de **offset** .

```
Floor (x[, step[, offset]])
```

round

Round() retorna o resultado do arredondamento de um número para cima ou para baixo, para o múltiplo mais próximo da offset **step** deslocada pelo **offset** número de

```
Round ( x [ , step [ , offset ]])
```

BitCount

BitCount() retorna quantos bits no equivalente binário de um número decimal estão definidos como 1. Ou seja, a função retorna o número de bits definidos em **integer_number**, em que **integer_number** é interpretado como um número inteiro de 32 bits com sinal.

Sintaxe:

```
BitCount(integer_number)
```

Tipo de dados de retorno: inteiro

Exemplos e resultados:

Exemplos e resultados

Exemplos	Resultados
`BitCount ( 3 )`	3 é binário 11 e, portanto, retorna 2
`BitCount ( -1 )`	-1 é 64 uns em um binário; portanto, ele retorna 64

Ceil

Ceil() arredonda um número para o múltiplo mais próximo da **step** deslocada pelo número de **offset** .

Compare com a função **floor**, que arredonda os números da entrada para baixo.

Sintaxe:

```
Ceil(x[, step[, offset]])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
x	Número de entrada.
step	Incremento do intervalo. O valor padrão é 1.
offset	Define a base do intervalo da etapa. O valor padrão é 0.

Exemplos e resultados:

Exemplos e resultados

Exemplos	Resultados
`Ceil(2.4 )`	Retorna 3 Neste exemplo, o tamanho da etapa é 1 e a base do intervalo da etapa é 0. Os intervalos são ...0 < x <=1, 1 < x <= 2, **2< x <=3**, 3< x <=4...
`Ceil(4.2 )`	Retorna 5
`Ceil(3.88 ,0.1)`	Retorna 3,9 Neste exemplo, o tamanho do intervalo é 0,1 e a base do intervalo é 0. Os intervalos são ... 3.7 < x <= 3.8, **3.8 < x <= 3.9**, 3.9 < x <= 4.0...
`Ceil(3.88 ,5)`	Retorna 5
`Ceil(1.1 ,1)`	Retorna 2
`Ceil(1.1 ,1,0.5)`	Retorna 1,5 Neste exemplo, o tamanho da etapa é 1 e o deslocamento é 0,5. Isso significa que a base do intervalo da etapa é 0,5, e não 0. Os intervalos são ...**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)`	Retorna 1,99 Os intervalos são ...-0.01< x <= 0.99, **0.99< x <= 1.99**, 1.99 < x <=2.99...

Combin

Combin() retorna o número de combinações de elementos **q** que podem ser coletados de um grupo de itens **p**. Conforme representado pela fórmula: `Combin(p,q) = p! / q!(p-q)!` A ordem na qual os itens são selecionados é insignificante.

Sintaxe:

```
Combin(p, q)
```

Tipo de dados de retorno: inteiro

Limitações:

Os itens não inteiros serão truncados.

Exemplos e resultados:

Exemplos e resultados

Exemplos	Resultados
Quantas combinações de 7 números podem ser selecionadas de um total de 35 números da loteria? `Combin( 35,7 )`	Retorna 6.724.520

Div

Div() retorna a parte inteira da divisão aritmética do primeiro argumento pelo segundo argumento. Os dois parâmetros são interpretados como números reais, isto é, não precisam ser inteiros.

Sintaxe:

```
Div(integer_number1, integer_number2)
```

Tipo de dados de retorno: inteiro

Exemplos e resultados:

Exemplos e resultados

Exemplos	Resultados
`Div( 7,2 )`	Retorna 3
`Div( 7.1,2.3 )`	Retorna 3
`Div( 9,3 )`	Retorna 3
`Div( -4,3 )`	Retorna -1
`Div( 4,-3 )`	Retorna -1
`Div( -4,-3 )`	Retorna 1

Even

Even() retorna True (-1), se **integer_number** for um número inteiro par ou zero. Retorna False (0), se **integer_number** for um número inteiro ímpar, e NULL se **integer_number** não for um número inteiro.

Sintaxe:

```
Even(integer_number)
```

Tipo de dados de retorno: Booleano

Exemplos e resultados:

Exemplos e resultados

Exemplos	Resultados
`Even( 3 )`	Retorna 0, False
`Even( 2 * 10 )`	Retorna -1, True
`Even( 3.14 )`	Retorna NULL

Fabs

Fabs() retorna o valor absoluto de **x**. O resultado é um número positivo.

Sintaxe:

```
fabs(x)
```

Tipo de dados de retorno: numérico

Exemplos e resultados:

Exemplos e resultados

Exemplos	Resultados
`fabs( 2.4 )`	Retorna 2,4
`fabs( -3.8 )`	Retorna 3,8

Fact

Fact() retorna o fatorial de um inteiro positivo **x**.

Sintaxe:

```
Fact(x)
```

Tipo de dados de retorno: inteiro

Limitações:

Se o número **x** não for um inteiro, ele será truncado. Os números não positivos retornarão NULL.

Exemplos e resultados:

Exemplos e resultados

Exemplos	Resultados
`Fact( 1 )`	Retorna 1
`Fact( 5 )`	Retorna 120 (1 * 2 * 3 * 4 * 5 = 120)
`Fact( -5 )`	Retorna NULL

Floor

Floor() arredonda um número para baixo, para o múltiplo mais próximo da **step** deslocada pelo número de **offset** .

Compare com a função **ceil**, que arredonda os números de entrada para cima.

Sintaxe:

```
Floor(x[, step[, offset]])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
x	Número de entrada.
step	Incremento do intervalo. O valor padrão é 1.
offset	Define a base do intervalo da etapa. O valor padrão é 0.

Exemplos e resultados:

Exemplos e resultados

Exemplos	Resultados
`Floor(2.4)`	Retorna 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)`	Retorna 4

Exemplos	Resultados
`Floor(3.88 ,0.1)`	Retorna 3,8 Neste exemplo, o tamanho do intervalo é 0,1 e a base do intervalo é 0. Os intervalos são ... 3.7 <= x < 3.8, **3.8 <= x < 3.9**, 3.9 <= x < 4.0...
`Floor(3.88 ,5)`	Retorna 0
`Floor(1.1 ,1)`	Retorna 1
`Floor(1.1 ,1,0.5)`	Retorna 0,5 Neste exemplo, o tamanho da etapa é 1 e o deslocamento é 0,5. Isso significa que a base do intervalo da etapa é 0,5, e não 0. Os intervalos são ...**0.5 <= x <1.5**, 1.5 <= x < 2.5, 2.5<= x <3.5,...

Fmod

fmod() é uma função de módulo generalizada que retorna a parte restante da divisão de inteiro do primeiro argumento (o dividendo) pelo segundo argumento (o divisor). O resultado é um número real. Os dois argumentos são interpretados como números reais, isto é, não precisam ser inteiros.

Sintaxe:

```
fmod(a, b)
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
a	Dividendo
b	Divisor

Exemplos e resultados:

Exemplos e resultados

Exemplos	Resultados
`fmod( 7,2 )`	Retorna 1
`fmod( 7.5,2 )`	Retorna 1,5
`fmod( 9,3 )`	Retorna 0
`fmod( -4,3 )`	Retorna -1
`fmod( 4,-3 )`	Retorna 1
`fmod( -4,-3 )`	Retorna -1

Frac

Frac() retorna a parte fracionária de **x**.

A fração é definida de modo que `Frac(x ) + Floor(x ) = x`. Em termos simples, isso significa que a parte fracionária de um número positivo é a diferença entre o número (x) e o número inteiro que precede a parte fracionária.

Por exemplo: A parte fracional de 11,43 = 11,43 - 11 = 0,43

Para um número negativo, digamos -1,4, Floor(-1.4) = -2, que gera o seguinte resultado:

A parte fracionada de -1,4 = -1,4 - (-2) = -1,4 + 2 = 0,6

Sintaxe:

```
Frac(x)
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
x	Número do qual retornar a fração.

Exemplos e resultados:

Exemplos e resultados

Exemplos	Resultados
`Frac( 11.43 )`	Retorna 0,43
`Frac( -1.4 )`	Retorna 0,6
Extrai o componente de hora da representação numérica de um carimbo de data/hora, omitindo assim a data. `Time(Frac(44518.663888889))`	Retorna 3:56:00 PM

Mod

Mod() é uma função de módulo matemático que retorna o restante não negativo de uma divisão de inteiro. O primeiro argumento é o dividendo e o segundo é o divisor. Os dois argumentos devem ser valores inteiros.

Sintaxe:

```
Mod(integer_number1, integer_number2)
```

Tipo de dados de retorno: inteiro

Limitações:

integer_number2 deve ser maior que 0.

Exemplos e resultados:

Exemplos e resultados

Exemplos	Resultados
`Mod( 7,2 )`	Retorna 1
`Mod( 7.5,2 )`	Retorna NULL
`Mod( 9,3 )`	Retorna 0
`Mod( -4,3 )`	Retorna 2
`Mod( 4,-3 )`	Retorna NULL
`Mod( -4,-3 )`	Retorna NULL

Odd

Odd() retorna True (-1), se **integer_number** for um número inteiro ímpar ou zero. Retorna False (0), se **integer_number** for um número inteiro par, e NULL se **integer_number** não for um número inteiro.

Sintaxe:

```
Odd(integer_number)
```

Tipo de dados de retorno: Booleano

Exemplos e resultados:

Exemplos e resultados

Exemplos	Resultados
`Odd( 3 )`	Retorna -1, True
`Odd( 2 * 10 )`	Retorna 0, False
`Odd( 3.14 )`	Retorna NULL

Permut

Permut() retorna o número de permutas de elementos **q** que podem ser selecionadas de um grupo de itens **p**. Conforme representado pela fórmula: `Permut(p,q) = (p)! / (p - q)!` A ordem na qual os itens são selecionados é significativa.

Sintaxe:

```
Permut(p, q)
```

Tipo de dados de retorno: inteiro

Limitações:

Os argumentos não inteiros serão truncados.

Exemplos e resultados:

Exemplos e resultados

Exemplos	Resultados
De quantas maneiras diferentes é possível distribuir as medalhas de ouro, de prata e de bronze após um final de 100 m com 8 participantes? `Permut( 8,3 )`	Retorna 336

Round

Round() retorna o resultado do arredondamento de um número para cima ou para baixo, para o múltiplo mais próximo da offset **step** deslocada pelo **offset** número de

Se o número para arredondar estiver exatamente no meio de um intervalo, será arredondado para cima.

Sintaxe:

```
Round(x[, step[, offset]])
```

Tipo de dados de retorno: numérico

Se você estiver arredondando um número de ponto flutuante, poderá observar resultados incorretos. Esses erros de arredondamento ocorrem porque números de ponto flutuante são representados por um número finito de dígitos binários. Portanto, os resultados são calculados usando um número que já foi arredondado. Se esses erros de arredondamento afetarem seu trabalho, multiplique os números para convertê-los em números inteiros, antes do arredondamento.

Argumentos:

Argumentos

Argumento	Descrição
x	Número de entrada.
step	Incremento do intervalo. O valor padrão é 1.
offset	Define a base do intervalo da etapa. O valor padrão é 0.

Exemplos e resultados:

Exemplos e resultados

Exemplos	Resultados
`Round(3.8 )`	Retorna 4 Neste exemplo, o tamanho da etapa é 1 e a base do intervalo da etapa é 0. Os intervalos são ...0 <= x <1, 1 <= x < 2, 2<= x <3, **3<= x <4**...
`Round(3.8,4 )`	Retorna 4
`Round(2.5 )`	Retorna 3. Neste exemplo, o tamanho da etapa é 1 e a base do intervalo da etapa é 0. Os intervalos são ...0 <= x <1, 1 <= x <2, **2<= x <3**...
`Round(2,4 )`	Retorna 4. Arredondado, pois 2 é exatamente a metade do intervalo da etapa de 4. Neste exemplo, o tamanho da etapa é 4 e a base do intervalo da etapa é 0. Os intervalos são ...**0 <= x <4**, 4 <= x <8, 8<= x <12...
`Round(2,6 )`	Retorna 0. Arredondado, pois 2 é exatamente a metade do intervalo da etapa de 6. Neste exemplo, o tamanho da etapa é 6 e a base do intervalo da etapa é 0. Os intervalos são ...**0 <= x <6**, 6 <= x <12, 12<= x <18...
`Round(3.88 ,0.1)`	Retorna 3,9 Neste exemplo, o tamanho da etapa é 0,1 e a base do intervalo da etapa é 0. Os intervalos são ... 3.7 <= x <3.8, **3.8 <= x <3.9**, 3.9 <= x < 4.0...
`Round (3.88875,1/1000)`	Retorna 3,889 Neste exemplo, o tamanho da etapa é 0,001, o que arredonda o número para cima e o limita a três casas decimais.
`Round(3.88 ,5)`	Retorna 5
`Round(1.1 ,1,0.5)`	Retorna 1,5 Neste exemplo, o tamanho da etapa é 1 e a base do intervalo da etapa é 0,5. Os intervalos são ...**0.5 <= x <1.5**, 1.5 <= x <2.5, 2.5<= x <3.5...

Sign

Sign() retorna 1, 0 ou -1, dependendo se **x** for um número positivo, 0 ou um número negativo.

Sintaxe:

```
Sign(x)
```

Tipo de dados de retorno: numérico

Limitações:

Se não for encontrado nenhum valor numérico, será retornado NULL.

Exemplos e resultados:

Exemplos e resultados

Exemplos	Resultados
`Sign( 66 )`	Retorna 1
`Sign( 0 )`	Retorna 0
`Sign( - 234 )`	Retorna -1

5.14 Funções geoespaciais

Estas funções são utilizadas para manusear dados geoespaciais nas visualizações de mapa. Qlik Sense segue as especificações GeoJSON para dados geoespaciais e oferece suporte ao seguinte:

- Ponto
- String de linha
- Polígono
- Multipolígono

Para obter mais informações sobre especificações GeoJSON, consulte:
GeoJSON.org

Visão geral das funções geoespaciais

Cada função é descrita adicionalmente após a visão geral. Você também pode clicar no nome da função na sintaxe para acessar imediatamente os detalhes dessa função específica.

Essas são as duas categorias de funções geoespaciais: agregação e não agregação.

Funções de agregação pegam um conjunto de formas geométricas (pontos ou áreas) como entrada e retornam uma única forma geométrica. Por exemplo, múltiplas áreas podem ser fundidas e um único limite para a agregação pode ser traçado no mapa.

Funções de não agregação pegam uma única forma geométrica e retornam uma forma geométrica. Por exemplo, a função GeoGetPolygonCenter(), se o limite geográfico de uma área for ajustado como entrada, retorna o ponto geométrico (longitude e latitude) do centro daquela área.

As seguintes são funções de agregação:

GeoAggrGeometry

GeoAggrGeometry() é usada para agregar um número de áreas a uma área maior, por exemplo, agregando diversas sub-regiões em uma região.

```
GeoAggrGeometry (field_name)
```

GeoBoundingBox

GeoBoundingBox() é utilizada para agregar uma forma geométrica a uma área e para calcular a menor caixa de limite que contém todas as coordenadas.

```
GeoBoundingBox (field_name)
```

GeoCountVertex

GeoCountVertex() é usada para descobrir o número de vértices em um polígono.

```
GeoCountVertex(field_name)
```

GeoInvProjectGeometry

GeoInvProjectGeometry() é utilizada para agregar uma forma geométrica a uma área e aplicar o inverso de uma projeção.

```
GeoInvProjectGeometry(type, field_name)
```

GeoProjectGeometry

GeoProjectGeometry() é utilizada para agregar uma forma geométrica a uma área e aplicar uma projeção.

```
GeoProjectGeometry(type, field_name)
```

GeoReduceGeometry

GeoReduceGeometry() é usada para reduzir o número de vértices de uma geometria e para agregar diversas áreas em uma área, mas ainda exibir as linhas de limite das áreas individuais.

```
GeoReduceGeometry (geometry)
```

As seguintes são funções de não agregação:

GeoGetBoundingBox

GeoGetBoundingBox() é utilizada em scripts e expressões gráficas para calcular a menor caixa de limite geoespacial que contém todas as coordenadas de uma forma geométrica.

```
GeoGetBoundingBox (geometry)
```

GeoGetPolygonCenter

GeoGetPolygonCenter() é utilizada em scripts e expressões gráficas para calcular e retornar o ponto central de uma forma geométrica.

```
GeoGetPolygonCenter (geometry)
```

GeoMakePoint

GeoMakePoint() é utilizada em scripts e expressões gráficas para criar e marcar um ponto com latitude e longitude.

```
GeoMakePoint (lat_field_name, lon_field_name)
```

GeoProject

GeoProject() é utilizada em scripts e expressões gráficas para aplicar uma projeção a uma forma geométrica.

```
GeoProject (type, field_name)
```

GeoAggrGeometry

GeoAggrGeometry() é usada para agregar um número de áreas a uma área maior, por exemplo, agregando diversas sub-regiões em uma região.

Sintaxe:

```
GeoAggrGeometry(field_name)
```

Tipo de dados de retorno: caractere

Argumentos:

Argumentos

Argumento	Descrição
field_name	Um campo ou expressão que faça referência a um campo contendo a forma geométrica a ser representada. Isso pode ser um ponto (ou conjunto de pontos) fornecendo longitude e latitude, ou uma área.

Geralmente, **GeoAggrGeometry()** pode ser usado para combinar os dados de limite geoespacial. Por exemplo, você pode ter áreas de códigos postais para subúrbios em uma cidade e receitas de vendas para cada área. Se o território de um vendedor abrange áreas de diversos CEPs, pode ser útil apresentar as vendas totais por território de vendas, ao invés de por áreas individuais, e exibir os resultados em um mapa colorido.

GeoAggrGeometry() pode calcular a agregação das formas geométricas do subúrbio específico e gerar a forma geométrica do território fundida no modelo de dados. Se, então, os limites do território de vendas forem ajustados, quando os dados forem recarregados, os novos limites e receitas fundidos estarão refletidos no mapa.

Como **GeoAggrGeometry()** é uma função de agregação, se você usá-la no script, será necessário um comando **LOAD** com uma cláusula **Group by**.

*As linhas de limite de mapas criadas usando **GeoAggrGeometry()** são aquelas das áreas fundidas. Caso queira exibir as linhas de limite individuais das áreas pré-agregadas, use **GeoReduceGeometry ()**.*

Exemplos:
Este exemplo carrega um arquivo KML com os dados da área, e depois carrega uma tabela com os dados da área agregada.

```
[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() é utilizada para agregar uma forma geométrica a uma área e para calcular a menor caixa de limite que contém todas as coordenadas.

Uma GeoBoundingBox é representada como uma lista de quatro valores: esquerda, direita, parte superior, parte inferior.

Sintaxe:

```
GeoBoundingBox(field_name)
```

Tipo de dados de retorno: caractere

Argumentos:

Argumentos

Argumento	Descrição
field_name	Um campo ou expressão que faça referência a um campo contendo a forma geométrica a ser representada. Isso pode ser um ponto (ou conjunto de pontos) fornecendo longitude e latitude, ou uma área.

GeoBoundingBox() agrega um conjunto de formas geométricas e retorna quatro coordenadas para o menor retângulo que contém todas as coordenadas da forma geométrica agregada.

Para visualizar o resultado em um mapa, transfira a string resultante de quatro coordenadas em um formato de polígono, marque o campo transferido com um formato geopoligonal, arraste e solte esse campo no objeto do mapa. As caixas retangulares será exibida na visualização do mapa.

GeoCountVertex

GeoCountVertex() é usada para descobrir o número de vértices em um polígono.

Sintaxe:

```
GeoCountVertex(field_name)
```

Tipo de dados de retorno: inteiro

Argumentos:

Argumentos

Argumento	Descrição
field_name	Um campo ou expressão que faça referência a um campo contendo a forma geométrica a ser representada. Isso pode ser um ponto (ou conjunto de pontos) fornecendo longitude e latitude, ou uma área.

GeoGetBoundingBox

GeoGetBoundingBox() é utilizada em scripts e expressões gráficas para calcular a menor caixa de limite geoespacial que contém todas as coordenadas de uma forma geométrica.

Uma caixa de limite geoespacial, criada pela função GeoBoundingBox() é representada como uma lista de quatro valores: esquerda direita, parte superior, parte inferior.

Sintaxe:

```
GeoGetBoundingBox(field_name)
```

Tipo de dados de retorno: caractere

Argumentos:

Argumentos

Argumento	Descrição
field_name	Um campo ou expressão que faça referência a um campo contendo a forma geométrica a ser representada. Isso pode ser um ponto (ou conjunto de pontos) fornecendo longitude e latitude, ou uma área.

*Não use a cláusula **Group by** no editor de carga de dados com essa e outras funções geoespaciais não agregadoras, pois gerará um erro de carga.*

GeoGetPolygonCenter

GeoGetPolygonCenter() é utilizada em scripts e expressões gráficas para calcular e retornar o ponto central de uma forma geométrica.

Em alguns casos, é necessário plotar um ponto em vez de de um preenchimento de cor em um mapa. Se os dados geoespaciais existentes estiverem disponíveis apenas na forma geométrica da área (por exemplo, como um limite), use **GeoGetPolygonCenter()** para recuperar um par longitude e latitude para o centro da área.

Sintaxe:

```
GeoGetPolygonCenter(field_name)
```

Tipo de dados de retorno: caractere

Argumentos:

Argumentos

Argumento	Descrição
field_name	Um campo ou expressão que faça referência a um campo contendo a forma geométrica a ser representada. Isso pode ser um ponto (ou conjunto de pontos) fornecendo longitude e latitude, ou uma área.

*Não use a cláusula **Group by** no editor de carga de dados com essa e outras funções geoespaciais não agregadoras, pois gerará um erro de carga.*

GeoInvProjectGeometry

GeoInvProjectGeometry() é utilizada para agregar uma forma geométrica a uma área e aplicar o inverso de uma projeção.

Sintaxe:

```
GeoInvProjectGeometry(type, field_name)
```

Tipo de dados de retorno: caractere

Argumentos:

Argumentos

Argumento	Descrição
type	Tipo de projeção usado na transformação da forma geométrica do mapa. O que pode assumir um dos dois valores: 'unit', (default), que resulta em uma projeção 1:1 ou 'mercator', que usa a projeção Mercator padrão.
field_name	Um campo ou expressão que faça referência a um campo contendo a forma geométrica a ser representada. Isso pode ser um ponto (ou conjunto de pontos) fornecendo longitude e latitude, ou uma área.

Exemplo:

Exemplo de scripts

Exemplo	Resultado
Em um comando `Load`: `GeoInvProjectGeometry ('mercator',AreaPolygon) as InvProjectGeometry`	A forma geométrica carregada como **AreaPolygon** é transformada usando a transformação inversa da projeção Mercator e armazenada como **InvProjectGeometry** para uso em visualizações.

GeoMakePoint

GeoMakePoint() é utilizada em scripts e expressões gráficas para criar e marcar um ponto com latitude e longitude. GeoMakePoint retorna pontos na ordem da longitude e da latitude.

Sintaxe:

```
GeoMakePoint(lat_field_name, lon_field_name)
```

Tipo de dados de retorno: string, formatada [longitude, latitude]

Argumentos:

Argumentos

Argumento	Descrição
lat_field_name	Um campo ou expressão que se refere a um campo que representa a latitude do ponto.
lon_field_name	Um campo ou expressão que se refere a um campo que representa a longitude do ponto.

*Não use a cláusula **Group by** no editor de carga de dados com essa e outras funções geoespaciais não agregadoras, pois gerará um erro de carga.*

GeoProject

GeoProject() é utilizada em scripts e expressões gráficas para aplicar uma projeção a uma forma geométrica.

Sintaxe:

```
GeoProject(type, field_name)
```

Tipo de dados de retorno: caractere

Argumentos:

Argumentos

Argumento	Descrição
type	Tipo de projeção usado na transformação da forma geométrica do mapa. O que pode assumir um dos dois valores: 'unit', (default), que resulta em uma projeção 1:1 ou 'mercator', que usa a projeção Mercator da web.
field_name	Um campo ou expressão que faça referência a um campo contendo a forma geométrica a ser representada. Isso pode ser um ponto (ou conjunto de pontos) fornecendo longitude e latitude, ou uma área.

*Não use a cláusula **Group by** no editor de carga de dados com essa e outras funções geoespaciais não agregadoras, pois gerará um erro de carga.*

Exemplo:

Exemplos de script

Exemplo	Resultado
Em um comando `Load`: `GeoProject ('mercator',Area) as GetProject`	A projeção Mercator é aplicada à forma geométrica carregada como **Area** e o resultado é armazenado como **GetProject**.

GeoProjectGeometry

GeoProjectGeometry() é utilizada para agregar uma forma geométrica a uma área e aplicar uma projeção.

Sintaxe:

```
GeoProjectGeometry(type, field_name)
```

Tipo de dados de retorno: caractere

Argumentos:

Argumentos

Argumento	Descrição
type	Tipo de projeção usado na transformação da forma geométrica do mapa. O que pode assumir um dos dois valores: 'unit', (default), que resulta em uma projeção 1:1 ou 'mercator', que usa a projeção Mercator da web.

Argumento	Descrição
field_name	Um campo ou expressão que faça referência a um campo contendo a forma geométrica a ser representada. Isso pode ser um ponto (ou conjunto de pontos) fornecendo longitude e latitude, ou uma área.

Exemplo:

Exemplo	Resultado
Em um comando `Load`: `GeoProjectGeometry ('mercator',AreaPolygon) as ProjectGeometry`	A forma geométrica carregada como **AreaPolygon** é transformada usando a projeção Mercator e armazenada como **ProjectGeometry** para uso em visualizações.

GeoReduceGeometry

GeoReduceGeometry() é usada para reduzir o número de vértices de uma geometria e para agregar diversas áreas em uma área, mas ainda exibir as linhas de limite das áreas individuais.

Sintaxe:

```
GeoReduceGeometry(field_name[, value])
```

Tipo de dados de retorno: caractere

Argumentos:

Argumentos

Argumento	Descrição
field_name	Um campo ou expressão que faça referência a um campo contendo a forma geométrica a ser representada. Isso pode ser um ponto (ou conjunto de pontos) fornecendo longitude e latitude, ou uma área.
value	A quantidade de redução a ser aplicada à geometria. O intervalo é de 0 a 1, com 0 representando nenhuma redução e 1 representando uma redução máxima dos vértices. *O uso de um valor value de 0,9 ou superior, com um conjunto de dados complexo, pode reduzir o número de vértices a um nível em que a representação visual é imprecisa.*

GeoReduceGeometry() também executa uma função semelhante a **GeoAggrGeometry()** porque agrega diversas áreas em uma área. A diferença é que as linhas individuais de limite dos dados pré-agregação são exibidos no mapa se você usar **GeoReduceGeometry()**.

Como **GeoReduceGeometry()** é uma função de agregação, se você usá-la no script, será necessário um comando **LOAD** com uma cláusula **Group by**.

Exemplos:
Este exemplo carrega um arquivo KML com os dados da área, e depois carrega uma tabela com os dados da área reduzida e agregada.

```
[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;
```

5.15 Funções de interpretação

As funções de interpretação avaliam os conteúdos dos campos ou expressões de texto de entrada e impõem formato de dados específicos no valor numérico resultante. Com essas funções, você pode especificar o formato do número de acordo com os tipos de dados, incluindo atributos, tais como: separador decimal, separador de milhar e formato de data.

As funções de interpretação retornam um valor duplo com os valores numéricos e da string, mas podem ser consideradas como a realização de uma conversão de string para número. As funções pegam o valor de texto da expressão de entrada e geram um número representando a string.

Em contraste, as funções de formato fazem o oposto: pegam as expressões numéricas e as avaliam como strings, especificando o formato de exibição do texto resultante.

Se nenhuma função de interpretação for usada, o Qlik Sense interpretará os dados como uma combinação de números, datas, horas, carimbos de data/hora e strings usando a configuração padrão de formato numérico, formato de data e formato de hora definidos pelas variáveis de script e pelo sistema operacional.

Todas as funções de interpretação podem ser usadas em scripts de carga de dados ou em expressões de gráfico.

Todas as representações numéricas são dadas com um ponto decimal como separador decimal.

Visão geral das funções de interpretação

Cada função é descrita adicionalmente após a visão geral. Você também pode clicar no nome da função na sintaxe para acessar imediatamente os detalhes dessa função específica.

Date#

Date# avalia uma expressão como uma data no formato especificado no segundo argumento, se fornecida. Se o código de formato não estiver especificado, será usado o formato de data padrão definido no sistema operacional.

```
Date# (page 1272)(text[, format])
```

Interval#

Interval#() avalia uma expressão de texto como um intervalo de tempo no formato definido no sistema operacional, por padrão, ou no formato especificado no segundo argumento, se fornecido.

```
Interval# (page 1273)(text[, format])
```

Money#

Money#() converte uma string de texto em um valor monetário, no formato definido no script de carregamento de dados ou no sistema operacional, a menos que uma string de formato seja fornecida. Símbolos de separadores decimal e de milhar personalizados são parâmetros opcionais.

```
Money# (page 1273)(text[, format[, dec_sep[, thou_sep ] ] ])
```

Num#

Num#() interpreta uma string de texto como um valor numérico, ou seja, converte a string de entrada em um número usando o formato especificado no segundo parâmetro. Se o segundo parâmetro for omitido, ele usará os separadores decimais e de milhar definidos no script de carregamento de dados. Símbolos de separadores decimais e de milhar personalizados são parâmetros opcionais.

```
Num# (page 1275)(text[ , format[, dec_sep[ , thou_sep]]])
```

Text

Text() força o tratamento da expressão como texto, mesmo que seja possível uma interpretação numérica.

```
Text(expr)
```

Time#

Time#() avalia uma expressão como um valor de tempo, no formato de tempo definido no script de carga de dados ou sistema operacional, a não ser que uma string de formato seja fornecida..

```
Time# (page 1276)(text[, format])
```

Timestamp#

Timestamp#() avalia uma expressão como um valor de tempo e data, no formato de carimbo de data e hora definido no script de carga de dados ou o sistema operacional, a não ser que uma string de formato seja fornecida.

```
Timestamp# (page 1277)(text[, format])
```

Consulte também:

- *Funções de formato (page 1235)*

Date#

Date# avalia uma expressão como uma data no formato especificado no segundo argumento, se fornecida.

Sintaxe:

```
Date#(text[, format])
```

Tipo de dados de retorno: dual

Argumentos:

Argumentos

Argumento	Descrição
text	A string de texto a ser avaliada.
format	Sequência descrevendo o formato da sequência de texto a ser avaliada. Se for omitido, será usado o formato de data definido nas variáveis do sistema no script de carregamento de dados ou no sistema operacional.

Exemplos e resultados:
Estes exemplos usam o formato de data **M/D/YYYY**. O formato de data é especificado no comando **SET DateFormat** na parte superior do script de carregamento de dados.

Adicione o script de exemplo ao seu aplicativo e execute-o.

```
Load *,

Num(Date#(StringDate)) as Date;

LOAD * INLINE [

StringDate

8/7/97

8/6/1997

]
```

Se você criar uma tabela com **StringDate** e **Date** como dimensões, os resultados serão os seguintes:

Resultados

StringDate	Date
8/7/97	35649
8/6/1997	35648

Interval#

Interval#() avalia uma expressão de texto como um intervalo de tempo no formato definido no sistema operacional, por padrão, ou no formato especificado no segundo argumento, se fornecido.

Sintaxe:

```
Interval#(text[, format])
```

Tipo de dados de retorno: dual

Argumentos:

Argumentos

Argumento	Descrição
text	A string de texto a ser avaliada.
format	A string que descreve o formato de entrada a ser usado para converter a sequência de caracteres para um intervalo numérico. Se omitida, serão utilizados os formatos de data abreviada, formato de hora e separador decimal configurados no sistema operacional.

A função **interval#** converte um intervalo de tempo de texto em um equivalente numérico.

Exemplos e resultados:
Os exemplos abaixo supõem as seguintes configurações do sistema operacional:

- Formato de data abreviada: YY-MM-DD
- Formato de hora: M/D/YY
- Separador de número decimal: .

Resultados

Exemplo	Resultado
`Interval#( A, 'D hh:mm' )` em que A='1 09:00'	1.375

Money#

Money#() converte uma string de texto em um valor monetário, no formato definido no script de carregamento de dados ou no sistema operacional, a menos que uma string de formato seja fornecida. Símbolos de separadores decimal e de milhar personalizados são parâmetros opcionais.

Sintaxe:

```
Money#(text[, format[, dec_sep [, thou_sep ] ] ])
```

Tipo de dados de retorno: dual

Argumentos:

Argumentos

Argumento	Descrição
text	A string de texto a ser avaliada.
format	A string que descreve o formato de entrada a ser usado para converter a sequência de caracteres para um intervalo numérico. Se omitida, será usado o formato monetário definido no sistema operacional.
dec_sep	String especificando o separador de número decimal. Se omitida, o valor MoneyDecimalSep definido no script de carregamento de dados será usado.
thou_sep	String especificando o separador de número milhar. Se omitida, o valor MoneyThousandSep definido no script de carregamento de dados será usado.

A função **money#** geralmente comporta-se como a função **num#**, mas adota como valores padrão de separadores decimais e de milhar as variáveis do script para formato de moeda ou a configuração do sistema para unidade monetária.

Exemplos e resultados:
Os exemplos abaixo supõem as duas seguintes configurações de sistema operacional:

- Configuração padrão do formato de moeda 1: kr # ##0,00
- Configuração padrão do formato de moeda 2: $ #,##0.00

```
Money#(A , '# ##0,00 kr' )
```
onde A=35 648,37 kr

Resultados

Resultados	Configuração 1	Configuração 2
String	35 648.37 kr	35 648.37 kr
Número	35648.37	3564837

```
Money#( A, ' $#', '.', ',' )
```
onde A= $35,648.37

Resultados

Resultados	Configuração 1	Configuração 2
String	$35,648.37	$35,648.37
Número	35648.37	35648.37

Num#

Num#() interpreta uma string de texto como um valor numérico, ou seja, converte a string de entrada em um número usando o formato especificado no segundo parâmetro. Se o segundo parâmetro for omitido, ele usará os separadores decimais e de milhar definidos no script de carregamento de dados. Símbolos de separadores decimais e de milhar personalizados são parâmetros opcionais.

Sintaxe:

```
Num#(text[, format[, dec_sep [, thou_sep ] ] ])
```

Tipo de dados de retorno: dual

A função **Num#()** retorna um valor duplo como a string e o valor numérico. A função usa a representação textual da expressão de entrada e gera um número. Ela não altera o formato do número: a saída é formatada da mesma forma que a entrada.

Argumentos:

Argumentos

Argumento	Descrição
text	A string de texto a ser avaliada.
format	String que especifica o formato de número usado no primeiro parâmetro. Se omitida, os separadores decimais e de milhar definidos no script de carregamento de dados serão usados.
dec_sep	String que especifica o separador de número decimal. Se omitida, o valor da variável DecimalSep definido no script de carregamento de dados será usado.
thou_sep	String que especifica o separador de número milhar. Se omitida, o valor da variável ThousandSep definido no script de carregamento de dados será usado.

Exemplos e resultados:
A tabela a seguir mostra o resultado de *Num#(A, '#', '.' , ',')* para diferentes valores de A.

Resultados

A	Representação de string	Valor numérico (aqui exibido com casa decimal)
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() força o tratamento da expressão como texto, mesmo que seja possível uma interpretação numérica.

Sintaxe:

```
Text (expr)
```

Tipo de dados de retorno: dual

Exemplo:

```
Text( A )
```
em que A=1234

Resultados

String	Número
1234	-

Exemplo:

```
Text( pi( ) )
```

Resultados

String	Número
3.1415926535898	-

Time#

Time#() avalia uma expressão como um valor de tempo, no formato de tempo definido no script de carga de dados ou sistema operacional, a não ser que uma string de formato seja fornecida..

Sintaxe:

```
time#(text[, format])
```

Tipo de dados de retorno: dual

Argumentos:

Argumentos

Argumento	Descrição
text	A string de texto a ser avaliada.
format	Sequência descrevendo o formato da sequência de texto a ser avaliada. Se omitida, serão utilizados formatos de data abreviada, formato de hora e separador decimal configurados no sistema operacional.

Exemplo:

- Configuração padrão do formato de tempo 1: hh:mm:ss
- Configuração padrão do formato de tempo 2: hh.mm.ss

```
time#( A )
```
onde A=09:00:00

Resultados

Resultados	Configuração 1	Configuração 2
String:	09:00:00	09:00:00
Número:	0.375	-

Exemplo:

- Configuração padrão do formato de tempo 1: hh:mm:ss
- Configuração padrão do formato de tempo 2: hh.mm.ss

```
time#( A, 'hh.mm' )
```
em que A=09,00

Resultados

Resultados	Configuração 1	Configuração 2
String:	09.00	09.00
Número:	0.375	0.375

Timestamp#

Timestamp#() avalia uma expressão como um valor de tempo e data, no formato de carimbo de data e hora definido no script de carga de dados ou o sistema operacional, a não ser que uma string de formato seja fornecida.

Sintaxe:

```
timestamp#(text[, format])
```

Tipo de dados de retorno: dual

Argumentos:

Argumentos

Argumento	Descrição
text	A string de texto a ser avaliada.

Argumento	Descrição
format	Sequência descrevendo o formato da sequência de texto a ser avaliada. Se omitida, serão utilizados formatos de data abreviada, formato de hora e separador decimal configurados no sistema operacional. A ISO 8601 é suportada para carimbos de data/hora.

Exemplo:

Estes exemplos usam o formato de data **M/D/YYYY**. O formato de data é especificado no comando **SET DateFormat** na parte superior do script de carregamento de dados.

Adicione o script de exemplo ao seu aplicativo e execute-o.

```
Load *,
Timestamp(Timestamp#(String)) as TS;
LOAD * INLINE [
String
2015-09-15T12:13:14
1952-10-16T13:14:00+0200
1109-03-01T14:15
];
```

Se você criar uma tabela com **String** e **TS** como dimensões, os resultados serão os seguintes:

Resultados

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

5.16 Funções interregistro

As funções inter-registro são usadas:

- No script de carga de dados, quando um valor dos registros anteriormente carregados é necessário para a avaliação do registro atual.
- Em uma expressão de gráfico, quando outro valor do conjunto de dados de uma visualização é necessário.

A classificação por valores y em gráficos ou por colunas de expressão em tabelas não é permitida quando uma função de gráfico entre registros é usada em qualquer uma das expressões do gráfico. Essas alternativas de classificação estão, portanto, automaticamente desabilitadas. Quando você usar uma função de gráfico entre registros em uma visualização ou tabela, a classificação da visualização será revertida para a entrada classificada da função entre registros. Essa limitação não se aplica à função de script equivalente, se houver uma.

Definições de expressão com auto-referência só podem ser feitas de maneira confiável em tabelas com menos de 100 linhas, mas isso pode variar dependendo do hardware no qual o mecanismo da Qlik está sendo executado.

Funções de linha

Essas funções só podem ser usadas em expressões de gráficos.

Above

Above() avalia uma expressão na linha acima da linha atual dentro de um segmento de coluna em uma tabela. A linha para a qual ela é calculada depende do valor de **offset**; se estiver presente, o padrão será a linha diretamente acima. Para gráficos que não sejam tabelas, **Above()** avalia a linha acima da linha atual, na tabela estática equivalente do gráfico.

```
Above - função de gráfico([TOTAL [<fld{,fld}>]] expr [ , offset [,count]])
```

Below

Below() avalia uma expressão na linha abaixo da linha atual dentro de um segmento de coluna em uma tabela. A linha para a qual ela é calculada depende do valor de **offset**; se estiver presente, o padrão será a linha diretamente inferior. Para gráficos que não sejam tabelas, **Below()** avalia a linha abaixo da coluna atual no equivalente de tabela estática do gráfico.

```
Below - função de gráfico([TOTAL[<fld{,fld}>]] expression [ , offset [,count
]])
```

Bottom

Bottom() avalia uma expressão na última linha (inferior) de um segmento de coluna em uma tabela. A linha para a qual ela é calculada depende do valor de **offset**, se presente, sendo o valor padrão a linha inferior. Para gráficos que não sejam tabelas, a avaliação é feita na última linha da coluna atual, no equivalente de tabela estática do gráfico.

```
Bottom - função de gráfico([TOTAL[<fld{,fld}>]] expr [ , offset [,count ]])
```

Top

Top() avalia uma expressão na primeira linha (superior) de um segmento de coluna em uma tabela. A linha para a qual ela é calculada depende do valor de **offset**; se estiver presente, o padrão será a linha superior. Para gráficos que não sejam tabelas, a avaliação **Top()** é feita na primeira linha da coluna atual, no equivalente de tabela estática do gráfico.

```
Top - função de gráfico([TOTAL [<fld{,fld}>]] expr [ , offset [,count ]])
```

NoOfRows

NoOfRows() retorna o número de linhas no atual segmento de coluna em uma tabela. Para gráficos de bitmap, **NoOfRows()** retorna o número de linhas no equivalente de tabela estática do gráfico.

```
NoOfRows - função de gráfico([TOTAL])
```

Funções de coluna

Essas funções só podem ser usadas em expressões de gráficos.

Column

Column() retorna o valor encontrado na coluna correspondente a **ColumnNo** em uma tabela simples, desconsiderando as dimensões. Por exemplo, **Column(2)** retorna o valor da segunda coluna de medida.

```
Column - função de gráfico(ColumnNo)
```

Dimensionality

Dimensionality() retorna o número de dimensões da linha atual. No caso de tabelas dinâmicas, a função que retorna o número total de colunas da dimensão que têm conteúdo não agregado, isto é, não contêm somas parciais ou agregações contraídas.

```
Dimensionality - função de gráfico ( )
```

Secondarydimensionality

SecondaryDimensionality() retorna o número de linhas da tabela dinâmica da dimensão que têm conteúdo não-agregado, isto é, não contêm somas parciais ou agregações contraídas. Essa função é a equivalente da função **dimensionality()** para dimensões horizontais de tabela dinâmica.

```
SecondaryDimensionality - função de gráfico ( )
```

Funções de campo

FieldIndex

FieldIndex() retorna a posição do valor de campo **value** encontrado no campo **field_name** (por ordem de carga).

```
FieldIndex (field_name , value)
```

FieldValue

FieldValue() retorna o valor de campo encontrado na posição **elem_no** do campo **field_name** (por ordem de carga).

```
FieldValue (field_name , elem_no)
```

FieldValueCount

FieldValueCount() é uma função de número **inteiro** que retorna o número de valores distintos em um campo.

```
FieldValueCount (field_name)
```

Funções de tabela dinâmica

Essas funções só podem ser usadas em expressões de gráficos.

After

After() retorna o valor de uma expressão avaliada com os valores de dimensão de uma tabela dinâmica como aparecerem na coluna depois da atual, em um segmento de linha da tabela dinâmica.

```
After - função de gráfico([TOTAL] expression [ , offset [,n]])
```

Before

Before() retorna o valor de uma expressão avaliada com os valores de dimensão de uma tabela dinâmica como aparecerem na coluna antes da atual, em um segmento de linha da tabela dinâmica.

```
Before - função de gráfico([TOTAL] expression [ , offset [,n]])
```

First

First() retorna o valor de uma expressão avaliada com os valores de dimensão de uma tabela dinâmica como aparecerem na primeira coluna do segmento de linha da tabela dinâmica. Esta função retorna NULL em todos os tipos de gráficos, exceto em tabelas dinâmicas.

```
First - função de gráfico([TOTAL] expression [ , offset [,n]])
```

Last

Last() retorna o valor de uma expressão avaliada com os valores de dimensão de uma tabela dinâmica como aparecerem na última coluna do segmento de linha da tabela dinâmica. Esta função retorna NULL em todos os tipos de gráficos, exceto em tabelas dinâmicas.

```
Last - função de gráfico([TOTAL] expression [ , offset [,n]])
```

ColumnNo

ColumnNo() retorna o número da coluna atual dentro do segmento da linha atual na tabela dinâmica. A primeira coluna é a de número 1.

```
ColumnNo - função de gráfico([TOTAL])
```

NoOfColumns

NoOfColumns() retorna o número de colunas no atual segmento da linha em uma tabela dinâmica.

```
NoOfColumns - função de gráfico([TOTAL])
```

Funções inter-registro no script de carga de dados

Exists

Exists() determina se um valor de campo específico já foi carregado no campo no script de carga de dados. A função retorna TRUE ou FALSE, de forma que pode ser usada na cláusula **where** de um comando **LOAD** ou um comando **IF**.

```
Exists (field_name [, expr])
```

LookUp

Lookup() examina uma tabela que já esteja carregada e retorna o valor de **field_name** correspondente à primeira ocorrência do valor **match_field_value** no campo **match_field_name**. A tabela pode ser a tabela atual ou outra tabela carregada anteriormente.

```
LookUp (field_name, match_field_name, match_field_value [, table_name])
```

Peek

Peek() retorna o valor de um campo em uma tabela para uma linha que já foi carregada. O número da linha pode ser especificado, assim como a tabela. Se nenhum número de linha for especificado, o último registro carregado anteriormente será usado.

```
Peek (field_name[, row_no[, table_name ] ])
```

Previous

Previous() encontra o valor da expressão **expr** usando dados do registro de entrada anterior que não foi descartado devido a uma cláusula **where**. No primeiro registro de uma tabela interna, a função retornará NULL.

```
Previous (page 1317)(expr)
```

Consulte também:

- *Funções de intervalo (page 1338)*

Above - função de gráfico

Above() avalia uma expressão na linha acima da linha atual dentro de um segmento de coluna em uma tabela. A linha para a qual ela é calculada depende do valor de **offset**; se estiver presente, o padrão será a linha diretamente acima. Para gráficos que não sejam tabelas, **Above()** avalia a linha acima da linha atual, na tabela estática equivalente do gráfico.

Sintaxe:

```
Above([TOTAL] expr [ , offset [,count]])
```

Tipo de dados de retorno: dual

Argumentos:

Argumentos

Argumento	Descrição
expr	A expressão ou campo que contém os dados a serem medidos.
offset	Especificar um offsetn, maior que 0 move a avaliação da expressão n para linhas acima da linha atual. A especificação de um deslocamento igual a 0 avaliará a expressão na linha atual. Especificando um número de compensação negativa faz a função Above funcionar como a função Below com o número de compensação positiva correspondente.
count	Ao especificar um terceiro argumento **count** maior que 1, a função retornará uma série de valores **count**, um para cada linha da tabela **count**, contado para cima a partir da célula original. Dessa forma, a função pode ser usada como um argumento para qualquer uma das funções de intervalo especiais. *Funções de intervalo (page 1338)*
TOTAL	Se a tabela for unidimensional ou se o qualificador **TOTAL** for usado como argumento, o segmento de coluna atual será sempre igual à coluna inteira.

Na primeira linha de um segmento de coluna, um valor NULL é retornado, já que não há uma linha acima dela.

Um segmento de coluna é definido como um subconjunto consecutivo de células que contêm os mesmos valores para as dimensões na ordem de classificação atual. As funções de inter-registro de gráficos são computadas no segmento de coluna, excluindo a dimensão mais à direita no gráfico de tabela simples equivalente. Se houver apenas uma dimensão no gráfico ou se o qualificador TOTAL for especificado, a expressão avalia a tabela completa.

Se a tabela ou o equivalente de tabela tiver várias dimensões verticais, o segmento de coluna atual incluirá somente linhas com os mesmos valores que a linha atual em todas as colunas de dimensão, exceto na coluna que mostrar a última dimensão na ordem de classificação entre os campos.

Limitações:

- As chamadas recursivas retornarão NULL.
- A classificação por valores y em gráficos ou por colunas de expressão em tabelas não é permitida quando essa função de gráfico é usada em qualquer uma das expressões do gráfico. Essas alternativas de classificação estão, portanto, automaticamente desabilitadas. Quando você usar essa função de gráfico em uma visualização ou tabela, a classificação da visualização será revertida para a entrada classificada dessa função.

Exemplos e resultados:

Example 1:

Visualização de tabela para o Exemplo 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

Na captura de tela da tabela mostrada neste exemplo, a visualização de tabela é criada da dimensão **Customer** e das medições `Sum(Sales)` e `Above(Sum(Sales))`.

A coluna Above(Sum(Sales)) retorna NULL para a linha **Customer** contendo **Astrida**, porque não há nenhuma linha acima dela. O resultado para a linha **Betacab** mostra o valor de Sum(Sales) para **Astrida**, o resultado para **Canutility** mostra o valor para **Sum(Sales)** para **Betacab** e assim por diante.

Para a coluna rotulada Sum(Sales)+Above(Sum(Sales)), a linha para **Betacab** mostra o resultado da soma dos valores **Sum(Sales)** para as linhas **Betacab** + **Astrida** (539+587). O resultado da linha **Betacab** mostra o resultado da soma dos valores **Sum(Sales)** para **Canutility** + **Canutility** (683+539).

A medição rotulada Above offset 3 criada usando a expressão `Sum(Sales)+Above(Sum(Sales), 3)` tem o argumento **offset**, definido como 3 e tem o efeito de obter o valor na linha três linhas acima da linha atual. Ele soma o valor **Sum(Sales)** para o atual **Customer** ao valor **Customer**três linhas acima. Os valores retornados para as três primeiras linhas **Customer** são null.

A tabela também mostra medidas mais complexas: uma criada a partir de `Sum(Sales)+Above(Sum(Sales))` e uma rotulada como **Higher?**, que é criada a partir de `IF(Sum(Sales)>Above(Sum(Sales)), 'Higher')`.

Essa função também pode ser usada para gráficos além de tabelas, como gráficos de barra.

Para outros tipos de gráficos, converta o gráfico para o equivalente de tabela simples, para que você possa interpretar facilmente a qual linha a função está relacionada.

Example 2:

Nas capturas de telas das tabelas mostradas neste exemplo, mais dimensões foram adicionadas às visualizações: **Month** e **Product**. Para gráficos com mais de uma dimensão, os resultados de expressões que contêm as funções **Above**, **Below**, **Top** e **Bottom** dependem da ordem em que as dimensões da coluna são classificadas pelo Qlik Sense. O Qlik Sense avalia as funções com base nos segmentos de coluna que resultam da dimensão que é classificada por último. A ordem de classificação da coluna é controlada no painel de propriedades em **Classificação** e não é necessariamente a ordem em que as colunas aparecem na tabela.

Na captura de tela a seguir da visualização de tabela do Exemplo 2, a última dimensão classificada é **Month**; portanto a função **Above** avalia com base em meses. Há uma série de resultados para cada valor **Product** para cada mês (**Jan** a **Aug**) – um segmento de coluna. Isso é seguido por uma série para o próximo segmento da coluna: para cada **Month** para o próximo **Product**. Haverá um segmento da coluna para cada valor **Customer** para cada **Product**.

Visualização de tabela para o Exemplo 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:

Na captura de tela da visualização de tabela para o Exemplo 3, a última dimensão classificada é **Product**. Isto é feito movendo a dimensão Product para a posição 3 na guia Classificação do painel de propriedades. A função **Above** é avaliada para cada **Product**, e como existem apenas dois produtos, **AA** e **BB**, há apenas um resultado não "null" em cada série. Na linha **BB** para o mês **Jan**, o valor para **Above(Sum(Sales))**, é 46. Para a linha **AA**, o valor é null. O valor em cada linha **AA** para qualquer mês será sempre null, porque não há valor de **Product** acima de AA. A segunda série é avaliada em**AA** e **BB** para o mês **Feb** para o valor **Customer**, **Astrida**. Quando todos os meses forem avaliados para **Astrida**, a sequência será repetida para o segundo **Customer**Betacab e assim por diante.

Visualização de tabela para o Exemplo 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

Exemplo 4

Example 4:	**Resultado**
A função Above pode ser usada como entrada para as funções range. Por exemplo: `RangeAvg (Above(Sum(Sales),1,3))`.	Nos argumentos da função Above(), offset é definido como 1 e count é definido como 3. A função encontra os resultados da expressão Sum(Sales) nas três linhas imediatamente acima da linha atual no segmento da coluna (onde há uma linha). Estes três valores são usados como entrada para a função RangeAvg(), que encontra a média de valores na variação de números fornecidos. Uma tabela com Customer como dimensão dá os seguintes resultados para a expressão RangeAvg(). Astrida - Betacab 587 Canutility 563 Divadip: 603

Dados usados nos exemplos:

```
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 '|');
```

Consulte também:

- *Below - função de gráfico (page 1286)*
- *Bottom - função de gráfico (page 1290)*
- *Top - função de gráfico (page 1319)*
- *RangeAvg (page 1341)*

Below - função de gráfico

Below() avalia uma expressão na linha abaixo da linha atual dentro de um segmento de coluna em uma tabela. A linha para a qual ela é calculada depende do valor de **offset**; se estiver presente, o padrão será a linha diretamente inferior. Para gráficos que não sejam tabelas, **Below()** avalia a linha abaixo da coluna atual no equivalente de tabela estática do gráfico.

Sintaxe:

```
Below([TOTAL] expr [ , offset [,count ]])
```

Tipo de dados de retorno: dual

Argumentos:

Argumentos

Argumento	Descrição
expr	A expressão ou campo que contém os dados a serem medidos.
offset	Especificar um **offset**n, maior que 1 move a avaliação da expressão para as linhas n abaixo da linha atual. A especificação de um deslocamento igual a 0 avaliará a expressão na linha atual. Especificando um número de compensação negativa faz a função **Below** funcionar como a função **Above** com o número de compensação positiva correspondente.
count	Ao especificar um terceiro parâmetro **count** maior que 1, a função retornará uma série de valores **count**, um para cada **count** linhas da tabela, contado para baixo da célula original. Dessa forma, a função pode ser usada como um argumento para qualquer uma das funções de intervalo especiais. *Funções de intervalo (page 1338)*
TOTAL	Se a tabela for unidimensional ou se o qualificador **TOTAL** for usado como argumento, o segmento de coluna atual será sempre igual à coluna inteira.

Na última linha de um segmento de coluna, um valor NULL é retornado, já que não há uma linha abaixo dela.

Um segmento de coluna é definido como um subconjunto consecutivo de células que contêm os mesmos valores para as dimensões na ordem de classificação atual. As funções de inter-registro de gráficos são computadas no segmento de coluna, excluindo a dimensão mais à direita no gráfico de tabela simples equivalente. Se houver apenas uma dimensão no gráfico ou se o qualificador TOTAL for especificado, a expressão avalia a tabela completa.

Se a tabela ou o equivalente de tabela tiver várias dimensões verticais, o segmento de coluna atual incluirá somente linhas com os mesmos valores que a linha atual em todas as colunas de dimensão, exceto na coluna que mostrar a última dimensão na ordem de classificação entre os campos.

Limitações:

- As chamadas recursivas retornarão NULL.
- A classificação por valores y em gráficos ou por colunas de expressão em tabelas não é permitida quando essa função de gráfico é usada em qualquer uma das expressões do gráfico. Essas alternativas de classificação estão, portanto, automaticamente desabilitadas. Quando você usar essa função de gráfico em uma visualização ou tabela, a classificação da visualização será revertida para a entrada classificada dessa função.

Exemplos e resultados:

Example 1:

Visualização de tabela para o Exemplo 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	-	-	-	-

Na tabela mostrada na captura de tela do Exemplo 1, a visualização de tabela é criada a partir da dimensão **Customer** e das medidas `Sum(Sales)` e `Below(Sum(Sales))`.

A coluna **Below(Sum(Sales))** retorna NULL para a linha **Customer** contendo **Divadip**, porque não há nenhuma linha abaixo dela. O resultado para a linha **Canutility** mostra o valor de Sum(Sales) para **Divadip**, o resultado para **Betacab** mostra o valor para **Sum(Sales)** para **Canutility** e assim por diante.

A tabela também mostra medidas mais complexas, que podem ser usadas nas colunas rotuladas: `Sum (Sales)+Below(Sum(Sales))`, **Below +Offset 3** e **Higher?**. Estas expressões funcionam conforme descrito nos parágrafos a seguir.

Para a coluna rotulada **Sum(Sales)+Below(Sum(Sales))**, a linha para **Betacab** mostra o resultado da soma dos valores **Sum(Sales)** para as linhas **Astrida** + **Astrida** (539+587). O resultado da linha **Betacab** mostra o resultado da soma dos valores **Sum(Sales)** para **Canutility** + **Betacab** (539+683).

A medição rotulada **Below +Offset 3** criada usando a expressão `Sum(Sales)+Below(Sum(Sales), 3)` tem o argumento **offset**, definido como 3 e tem o efeito de obter o valor na linha três linhas abaixo da linha atual. Ele soma o valor **Sum(Sales)** para o atual **Customer** ao valor **Customer**três linhas abaixo. Os valores para as três linhas inferiores **Customer** são null.

A medida rotulada **Higher?** é criada a partir da expressão:`IF(Sum(Sales)>Below(Sum(Sales)), 'Higher')`. Isto compara os valores da linha atual na medida **Sum(Sales)** com a linha abaixo dela. Se a linha atual for um valor maior, o texto "Higher" será a saída.

Essa função também pode ser usada para gráficos além de tabelas, como gráficos de barra.

Para outros tipos de gráficos, converta o gráfico para o equivalente de tabela simples, para que você possa interpretar facilmente a qual linha a função está relacionada.

Para gráficos com mais de uma dimensão, os resultados de expressões que contêm as funções **Above**, **Below**, **Top** e **Bottom** dependem da ordem em que as dimensões da coluna são classificadas pelo Qlik Sense. O Qlik Sense avalia as funções com base nos segmentos de coluna que resultam da dimensão que é classificada por

último. A ordem de classificação da coluna é controlada no painel de propriedades em **Classificação** e não é necessariamente a ordem em que as colunas aparecem na tabela.Consulte o exemplo: 2 na função **Above** para obter detalhes adicionais.

Exemplo 2

Example 2:	**Resultado**
A função **Below** pode ser usada como entrada para as funções range. Por exemplo: `RangeAvg (Below (Sum(Sales),1,3))`.	Nos argumentos da função **Below()**, offset é definido como 1 e count é definido como 3. A função encontra os resultados da expressão **Sum (Sales)** nas três linhas imediatamente abaixo da linha atual no segmento de coluna (onde há uma linha). Estes três valores são usados como entrada para a função RangeAvg(), que encontra a média de valores na variação de números fornecidos. Uma tabela com **Customer** como dimensão dá os seguintes resultados para a expressão RangeAvg().
	Astrida 659.67 Betacab 720 Canutility 757 Divadip: -

Dados usados nos exemplos:

```
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 '|');
```

Consulte também:

- *Above - função de gráfico (page 1282)*
- *Bottom - função de gráfico (page 1290)*
- *Top - função de gráfico (page 1319)*
- *RangeAvg (page 1341)*

Bottom - função de gráfico

Bottom() avalia uma expressão na última linha (inferior) de um segmento de coluna em uma tabela. A linha para a qual ela é calculada depende do valor de **offset**, se presente, sendo o valor padrão a linha inferior. Para gráficos que não sejam tabelas, a avaliação é feita na última linha da coluna atual, no equivalente de tabela estática do gráfico.

Sintaxe:

```
Bottom([TOTAL] expr [ , offset [,count ]])
```

Tipo de dados de retorno: dual

Argumentos:

Argumentos

Argumento	Descrição
expr	A expressão ou campo que contém os dados a serem medidos.
offset	Especificar um **offset**n maior que 1 move a avaliação da expressão para as linhas acima n da linha inferior. Especificando um número de compensação negativa faz a função **Bottom** funcionar como a função **Top** com o número de compensação positiva correspondente.
count	Ao especificar um terceiro parâmetro **count** maior que 1, a função retornará não um, mas uma série de valores **count**, um para cada uma das últimas linhas **count** do segmento da coluna atual. Dessa forma, a função pode ser usada como um argumento para qualquer uma das funções de intervalo especiais. *Funções de intervalo (page 1338)*
TOTAL	Se a tabela for unidimensional ou se o qualificador **TOTAL** for usado como argumento, o segmento de coluna atual será sempre igual à coluna inteira.

Um segmento de coluna é definido como um subconjunto consecutivo de células que contêm os mesmos valores para as dimensões na ordem de classificação atual. As funções de inter-registro de gráficos são computadas no segmento de coluna, excluindo a dimensão mais à direita no gráfico de tabela simples equivalente. Se houver apenas uma dimensão no gráfico ou se o qualificador TOTAL for especificado, a expressão avalia a tabela completa.

Se a tabela ou o equivalente de tabela tiver várias dimensões verticais, o segmento de coluna atual incluirá somente linhas com os mesmos valores que a linha atual em todas as colunas de dimensão, exceto na coluna que mostrar a última dimensão na ordem de classificação entre os campos.

Limitações:

- As chamadas recursivas retornarão NULL.
- A classificação por valores y em gráficos ou por colunas de expressão em tabelas não é permitida quando essa função de gráfico é usada em qualquer uma das expressões do gráfico. Essas alternativas de classificação estão, portanto, automaticamente desabilitadas. Quando você usar essa função de gráfico em uma visualização ou tabela, a classificação da visualização será revertida para a entrada classificada dessa função.

Exemplos e resultados:

Visualização de tabela para o Exemplo 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

Na captura de tela da tabela mostrada neste exemplo, a visualização de tabela é criada da dimensão **Customer** e das medições `Sum(Sales)` e `Bottom(Sum(Sales))`.

A coluna **Bottom(Sum(Sales))** retorna 757 para todas as linhas, porque esse é o valor da linha inferior: **Divadip**.

A tabela também mostra medições mais complexas: uma criada a partir de `Sum(Sales)+Bottom(Sum(Sales))` e uma rotulada como **Bottom offset 3**, que é criada usando a expressão `Sum(Sales)+Bottom(Sum(Sales), 3)` e tem o argumento **offset** definido como 3. Ela acrescenta o valor **Sum(Sales)** da linha atual ao valor da terceira linha a partir da linha inferior, isto é, a linha atual mais o valor de **Betacab**.

Exemplo: 2

Nas capturas de telas das tabelas mostradas neste exemplo, mais dimensões foram adicionadas às visualizações: **Month** e **Product**. Para gráficos com mais de uma dimensão, os resultados de expressões que contêm as funções **Above**, **Below**, **Top** e **Bottom** dependem da ordem em que as dimensões da coluna são classificadas pelo Qlik Sense. O Qlik Sense avalia as funções com base nos segmentos de coluna que resultam da dimensão que é classificada por último. A ordem de classificação da coluna é controlada no painel de propriedades em **Classificação** e não é necessariamente a ordem em que as colunas aparecem na tabela.

Na primeira tabela, a expressão é avaliada com base no **Month**, e na segunda tabela é avaliada com base no **Product**. A medição **End value** contém a expressão `Bottom(Sum(Sales))`. A linha inferior de **Month** é Dec e o valor de Dec dos dois valores de **Product** mostrados na captura de tela é 22. (Algumas linhas foram editadas da captura de tela para economia de espaço.)

Primeira tabela para o Exemplo 2. O valor de Bottom para a medida de End value baseada em 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

Segunda tabela para o Exemplo 2. O valor de Bottom para a medida de End value baseada em Product (BB para 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

Consulte o exemplo: 2 na função **Above** para obter detalhes adicionais.

Exemplo 3

Exemplo: 3	**Resultado**
A função **Bottom** pode ser usada como entrada para as funções range. Por exemplo: `RangeAvg (Bottom(Sum(Sales),1,3))`.	Nos argumentos para a função **Bottom()**, offset é definido como 1 e count é definido como 3. A função encontra os resultados da expressão **Sum (Sales)** em três linhas, com a linha acima da linha inferior no segmento de coluna (porque offset=1) e as duas linhas acima (onde há uma linha). Estes três valores são usados como entrada para a função RangeAvg(), que encontra a média de valores na variação de números fornecidos. Uma tabela com **Customer** como dimensão dá os seguintes resultados para a expressão 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 '|');
```

Consulte também:

Top - função de gráfico (page 1319)

Column - função de gráfico

Column() retorna o valor encontrado na coluna correspondente a **ColumnNo** em uma tabela simples, desconsiderando as dimensões. Por exemplo, **Column(2)** retorna o valor da segunda coluna de medida.

Sintaxe:

```
Column(ColumnNo)
```

Tipo de dados de retorno: dual

Argumentos:

Argumentos

Argumento	Descrição
ColumnNo	Número de uma coluna na tabela que contém uma medição. *A função Column() desconsidera colunas de dimensão.*

Limitações:

- As chamadas recursivas retornarão NULL.
- Se **ColumnNo** referenciar uma coluna para a qual não há nenhuma medição, um valor NULL é retornado.
- A classificação por valores y em gráficos ou por colunas de expressão em tabelas não é permitida quando essa função de gráfico é usada em qualquer uma das expressões do gráfico. Essas alternativas de classificação estão, portanto, automaticamente desabilitadas. Quando você usar essa função de gráfico em uma visualização ou tabela, a classificação da visualização será revertida para a entrada classificada dessa função.

Exemplos e resultados:

Exemplo: Porcentagem total de vendas

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

Exemplo: Porcentagem de vendas para clientes selecionados

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

Exemplos e resultados

Exemplos	Resultados
Order Value é adicionado à tabela como uma medição com a expressão: `Sum (UnitPrice*UnitSales)`. Total Sales Value é adicionado como uma medição com a expressão: `Sum(TOTAL UnitPrice*UnitSales)` % Sales é adicionado como uma medição com a expressão `100*Column(1)/Column(2)`	O resultado de Column(1) é obtido da coluna Order Value, porque essa é a primeira coluna de medição. O resultado de Column(2) é obtido de Total Sales Value, porque essa é a segunda coluna de medição. Consulte os resultados na coluna % Sales no exemplo *Porcentagem total de vendas (page 1294)*.
Faça a seleção Customer A.	A seleção altera Total Sales Value, e portanto %Sales. Consulte o exemplo *Porcentagem de vendas para clientes selecionados (page 1295)*.

Dados usados nos exemplos:

```
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 - função de gráfico

Dimensionality() retorna o número de dimensões da linha atual. No caso de tabelas dinâmicas, a função que retorna o número total de colunas da dimensão que têm conteúdo não agregado, isto é, não contêm somas parciais ou agregações contraídas.

Sintaxe:

```
Dimensionality ( )
```

Tipo de dados de retorno: inteiro

Limitações:

Essa função só está disponível para gráficos. Para todos os tipos de gráfico, exceto tabelas dinâmicas, ela retornará o número de dimensões em todas as linhas, exceto o total, que será 0.

A classificação por valores y em gráficos ou por colunas de expressão em tabelas não é permitida quando essa função de gráfico é usada em qualquer uma das expressões do gráfico. Essas alternativas de classificação estão, portanto, automaticamente desabilitadas. Quando você usar essa função de gráfico em uma visualização ou tabela, a classificação da visualização será revertida para a entrada classificada dessa função.

Exemplo: Expressão de gráfico usando Dimensionality

Exemplo: Expressão de gráfico
A função **Dimensionality()** pode ser usada com uma tabela dinâmica como uma expressão de gráfico onde você deseja aplicar uma formatação de célula diferente, dependendo do número de dimensões em uma linha que tenha dados não agregados. Este exemplo usa a função Dimensionality() para aplicar uma cor de fundo às células da tabela que correspondem a uma determinada condição.

Script de carregamento

Carregue os seguintes dados como um carregamento inline no editor de carregamento de dados para criar o exemplo de expressão de gráfico abaixo.

```
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 ',');
```

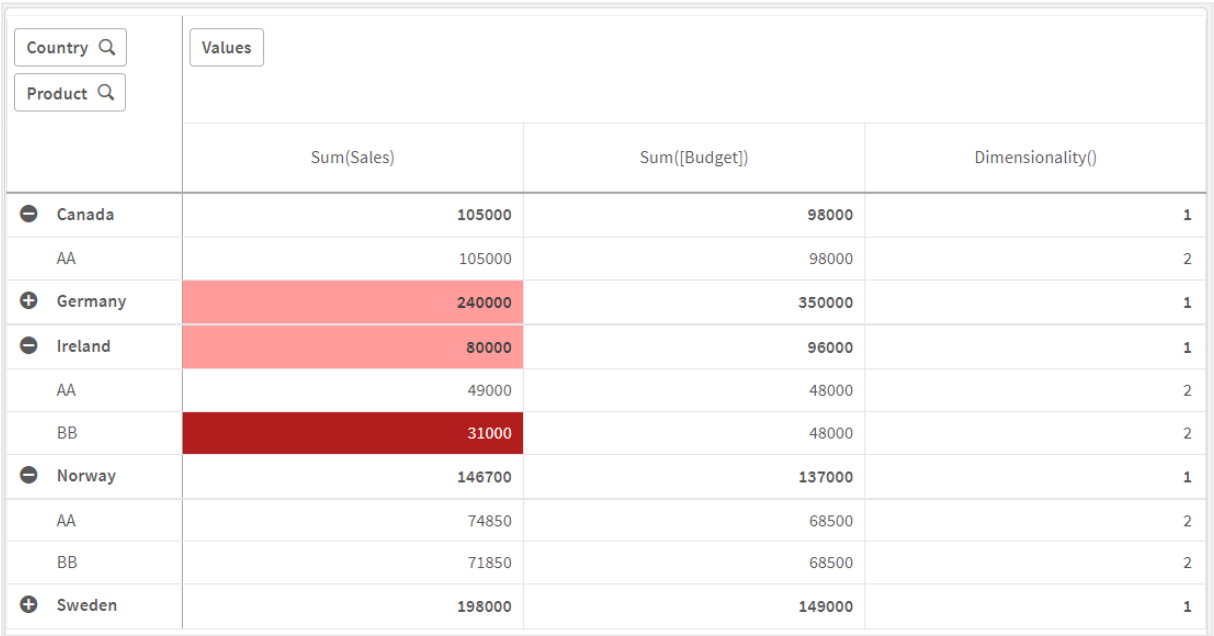
Expressão de gráfico

Crie uma visualização de tabela dinâmica em uma pasta do Qlik Sense com **País** e **Produto** como dimensões. Adicione **Sum(Sales)**, **Sum(Budget)** e **Dimensionality()** como medidas.

No painel **Propriedades**, insira a seguinte expressão como a **Expressão de cor de fundo** para a medida **Sum (Sales)**.

```
If(Dimensionality()=1 and Sum(Sales)<Sum(Budget),RGB(255,156,156),
If(Dimensionality()=2 and Sum(Sales)<Sum(Budget),RGB(178,29,29)
))
```

Resultado:



	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

Explicação

A expressão `If(Dimensionality()=1 and Sum(Sales)<Sum(Budget),RGB(255,156,156), If (Dimensionality()=2 and Sum(Sales)<Sum(Budget),RGB(178,29,29)))` contém declarações condicionais que verificam o valor da dimensionalidade e a Sum(Sales) e Sum(Budget) para cada produto. Se as condições forem atendidas, uma cor de fundo será aplicada ao valor de Sum(Sales).

Exists

Exists() determina se um valor de campo específico já foi carregado no campo no script de carga de dados. A função retorna TRUE ou FALSE, de forma que pode ser usada na cláusula **where** de um comando **LOAD** ou um comando **IF**.

*Também é possível usar **Not Exists()** para determinar se um valor de campo não foi carregado, mas recomenda-se cuidado se você usar **Not Exists()** em uma cláusula where. A função **Exists()** testa tanto as tabelas quanto os valores carregados anteriormente na tabela atual. Então, apenas a primeira ocorrência será carregada. Quando a segunda ocorrência é encontrada, o valor já está carregado. Veja os exemplos para mais informações.*

Sintaxe:

```
Exists(field_name [, expr])
```

Tipo de dados de retorno: Booleano

Argumentos:

Argumentos

Argumento	Descrição
field_name	O nome do campo onde você deseja procurar um valor. Você pode usar um nome de campo explícito sem aspas. O campo já deve estar carregado pelo script. Isso significa que você não pode se referir a um campo que seja carregado em uma cláusula mais abaixo no script.
expr	O valor que você deseja verificar se existe. Você pode usar um valor ou uma expressão explícita que se refira a um ou vários campos no comando load atual. *Você não pode se referir a campos que não estão incluídos no comando load atual.* Esse argumento é opcional. Se você omiti-lo, a função verificará se o valor de **field_name** no registro atual já existe.

Exemplos e resultados:

Exemplo 1

```
Exists (Employee)
```

Retorna -1 (True) se o valor do campo **Employee** no registro atual já existir em qualquer registro lido anteriormente que contenha esse campo.

Os comandos `Exists (Employee, Employee)` e `Exists (Employee)` são equivalentes.

Exemplo 2

```
Exists(Employee, 'Bill')
```

Retorna -1 (True) se o valor de campo **'Bill'** for encontrado no conteúdo atual do campo **Employee**.

Exemplo 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;
```

Isso resulta em uma tabela que pode ser visualizada em visualização de tabela usando as dimensões Employee e Address.

A cláusula where, `where Exists (Employee)`, significa que apenas os nomes da tabela Citizens que também estão em Employees são carregados na nova tabela. O comando Drop remove a tabela Employees para evitar confusão.

Resultados

Employee	Address
Bill	New York
John	Miami
Steve	Chicago

Exemplo 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;
```

A cláusula where inclui not: `where not Exists (Employee)`.

Isso significa que apenas os nomes da tabela Citizens que não estão em Employees são carregados para a nova tabela.

Observe que há dois valores para Lucy na tabela Citizens, mas apenas um é incluído na tabela de resultados. Quando você carrega a primeira linha com o valor Lucy, ela é incluída no campo Employee. Portanto, quando a segunda linha é verificada, o valor já existe.

Resultados

Funcionário	**Endereço**
Mary	London
Lucy	Madrid

Exemplo 5

Esse exemplo mostra como carregar todos os valores.

```
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;
```

Para obter todos os valores de Lucy, duas coisas foram alteradas:

- Um carregamento anterior na tabela Employees foi inserido no ponto em que Employee foi renomeado para Name.
  `Load Employee As Name;`
- A condição Onde em Citizens foi alterada para:
  `not Exists (Name, Employee).`

Isso cria campos para Name e Employee. Quando a segunda linha com Lucy está marcada, ela ainda não existe em Name.

Resultados

Funcionário	**Endereço**
Mary	London
Lucy	Madrid
Lucy	Paris

FieldIndex

FieldIndex() retorna a posição do valor de campo **value** encontrado no campo **field_name** (por ordem de carga).

Sintaxe:

```
FieldIndex(field_name , value)
```

Tipo de dados de retorno: inteiro

Argumentos:

Argumentos

Argumento	**Descrição**
field_name	Nome do campo para o qual é exigido o índice. Por exemplo, a coluna em uma tabela. Deve ser dado como um valor de caracteres. Isto significa que o nome do campo deve ser colocado entre aspas simples.
value	O valor do campo **field_name**.

Limitações:

- Se **value** valor não for encontrado entre os valores do campo **field_name**, 0 será retornado.
- A classificação por valores y em gráficos ou por colunas de expressão em tabelas não é permitida quando essa função de gráfico é usada em qualquer uma das expressões do gráfico. Essas alternativas de classificação estão, portanto, automaticamente desabilitadas. Quando você usar essa função de gráfico em uma visualização ou tabela, a classificação da visualização será revertida para a entrada classificada dessa função. Essa limitação não se aplica à função de script equivalente.

Exemplos e resultados:

Os exemplos a seguir usam o campo: **First name** da tabela**Names**.

Exemplos e resultados

Exemplos	**Resultados**
Adicione os dados de exemplo ao seu aplicativo e execute-o.	A tabela **Names** é carregada, como nos dados de amostra.
Função de gráfico: Em uma tabela contendo a dimensão First name, adicione como medida:	
`FieldIndex ('First name','John')`	1, porque 'John' aparece primeiro na ordem de carregamento do campo **First name**. Observe que em um painel de filtro, **John** apareceria como número 2 a partir do topo, pois é classificado em ordem alfabética e não na ordem de carregamento.
`FieldIndex ('First name','Peter')`	4, porque **FieldIndex()** retorna apenas um valor, que é a primeira ocorrência na ordem de carregamento.
Função de script: Dado que a tabela **Names** seja carregada, como nos dados de exemplo:	
`John1: Load FieldIndex('First name','John') as MyJohnPos Resident Names;`	`MyJohnPos=1`, porque 'John' aparece primeiro na ordem de carregamento do campo **First name**. Observe que em um painel de filtro, **John** apareceria como número 2 a partir do topo, pois é classificado em ordem alfabética e não na ordem de carregamento.
`Peter1: Load FieldIndex('First name','Peter') as MyPeterPos Resident Names;`	`MyPeterPos=4`, porque **FieldIndex()** retorna apenas um valor, que é a primeira ocorrência na ordem de carregamento.

Dados usados no exemplo:

```
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() retorna o valor de campo encontrado na posição **elem_no** do campo **field_name** (por ordem de carga).

Sintaxe:

```
FieldValue(field_name , elem_no)
```

Tipo de dados de retorno: dual

Argumentos:

Argumentos

Argumento	Descrição
field_name	Nome do campo para o qual é exigido o valor. Por exemplo, a coluna em uma tabela. Deve ser dado como um valor de caracteres. Isto significa que o nome do campo deve ser colocado entre aspas simples.
elem_no	O número da posição (elemento) do campo, seguindo a ordem da carga, para a qual o valor é retornado. Isso poderia corresponder à fileira na tabela, mas depende da ordem na qual os elementos (linhas) estiverem carregados.

Limitações:

- Se **elem_no** for maior que o número de valores de campo, será retornadoNULL.
- A classificação por valores y em gráficos ou por colunas de expressão em tabelas não é permitida quando essa função de gráfico é usada em qualquer uma das expressões do gráfico. Essas alternativas de classificação estão, portanto, automaticamente desabilitadas. Quando você usar essa função de gráfico em uma visualização ou tabela, a classificação da visualização será revertida para a entrada classificada dessa função. Essa limitação não se aplica à função de script equivalente.

Exemplo

Script de carregamento

Carregue os seguintes dados como um carregamento inline no editor de carregamento de dados para criar o exemplo abaixo.

```
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;
```

Criar uma visualização

Crie uma visualização de tabela em uma pasta do Qlik Sense. Adicione os campos **First name**, **MyPos1** e **MyPos2** à tabela.

Resultado

First name	MyPos1	MyPos2
Jane	John	Jane
John	John	Jane
Mark	John	Jane
Peter	John	Jane
Sue	John	Jane

Explicação

FieldValue('First name','1') resulta em John como o valor para **MyPos1** para todos os nomes porque John aparece primeiro na ordem de carregamento do campo **First name**. Observe que em um painel de filtro John apareceria como número 2 a partir do topo, depois de Jane, porque é classificado em ordem alfabética e não na ordem de carregamento.

FieldValue('First name','5') resulta em Jane como o valor para **MyPos2** para todos os nomes porque Jane aparece em quinto lugar na ordem de carregamento do campo **First name**.

FieldValueCount

FieldValueCount() é uma função de número **inteiro** que retorna o número de valores distintos em um campo.

Um carregamento parcial pode remover valores dos dados, o que não será refletido no número retornado. O número retornado corresponderá a todos os valores distintos que foram carregados no carregamento inicial ou em qualquer carregamento parcial subsequente.

A classificação por valores y em gráficos ou por colunas de expressão em tabelas não é permitida quando essa função de gráfico é usada em qualquer uma das expressões do gráfico. Essas alternativas de classificação estão, portanto, automaticamente desabilitadas. Quando você usar essa função de gráfico em uma visualização ou tabela, a classificação da visualização será revertida para a entrada classificada dessa função. Essa limitação não se aplica à função de script equivalente.

Sintaxe:

```
FieldValueCount(field_name)
```

Tipo de dados de retorno: inteiro

Argumentos:

Argumentos

Argumento	Descrição
field_name	Nome do campo para o qual é exigido o valor. Por exemplo, a coluna em uma tabela. Deve ser dado como um valor de caracteres. Isto significa que o nome do campo deve ser colocado entre aspas simples.

Exemplos e resultados:

Os exemplos a seguir usam o campo **First name** da tabela **Names**.

Exemplos e resultados

Exemplos	Resultados
Adicione os dados de exemplo ao seu aplicativo e execute-o.	A tabela **Names** é carregada, como nos dados de amostra.
Função de gráfico: Em uma tabela contendo a dimensão First name, adicione como medida:	
`FieldValueCount('First name')`	5 porque **Peter** aparece duas vezes.
`FieldValueCount('Initials')`	6 porque **Initials** tem apenas valores distintos.
Função de script: Dado que a tabela **Names** seja carregada, como nos dados de exemplo:	

`FieldCount1:`
`Load FieldValueCount('First name') as MyFieldCount1`
`Resident Names;`
	`MyFieldCount1`=5, porque 'Peter' aparece duas vezes.

`FieldCount2:`
`Load FieldValueCount('Initials') as MyInitialsCount1`
`Resident Names;`
	`MyFieldCount1`=6, porque 'Initials' tem apenas valores distintos.

Dados usados nos exemplos:

```
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() examina uma tabela que já esteja carregada e retorna o valor de **field_name** correspondente à primeira ocorrência do valor **match_field_value** no campo **match_field_name**. A tabela pode ser a tabela atual ou outra tabela carregada anteriormente.

Sintaxe:

```
lookup(field_name, match_field_name, match_field_value [, table_name])
```

Tipo de dados de retorno: dual

Argumentos:

Argumentos

Argumento	Descrição
field_name	Nome do campo para o qual o valor de retorno é exigido. O valor de entrada deve ser fornecido como uma string (por exemplo, literais entre aspas).
match_field_name	Nome do campo para procurar em **match_field_value**. O valor de entrada deve ser fornecido como uma string (por exemplo, literais entre aspas).
match_field_value	Valor para procurar no **campo** match_field_name.
table_name	Nome da tabela em que procurar o valor. O valor de entrada deve ser fornecido como uma string (por exemplo, literais entre aspas). Se **table_name** for omitido, a tabela atual será assumida.

Argumentos sem aspas, consulte a tabela atual. Para se referir a outras tabelas, coloque um argumento entre aspas simples.

Limitações:

A ordem da pesquisa é feita é na ordem de carga, a menos que a tabela seja o resultado de operações complexas, como uniões, em que a ordem não é bem definida. Tanto **field_name** quanto **match_field_name** devem ser campos da mesma tabela, especificados pelo **table_name**.

Se não for encontrada correspondência, NULL será retornado.

Exemplo

Script de carregamento

Carregue os seguintes dados como um carregamento inline no editor de carregamento de dados para criar o exemplo abaixo.

```
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;
```

Criar uma visualização

Crie uma visualização de tabela em uma Qlik Sense do Qlik Cloud. Adicione os campos **ProductID**, **InvoiceID**, **CustomerID**, **Units** e **CategoryID** à tabela.

Resultado

Tabela resultante

ProductID	InvoiceID	CustomerID	Unidades	CategoryID
1	1	Astrida	8	1
2	1	Astrida	6	1
3	2	Betacab	10	2

ProductID	InvoiceID	CustomerID	Unidades	CategoryID
3	3	Divadip	5	2
4	4	Divadip	10	3

Explicação

Os dados de amostra usam a função **Lookup()** da seguinte forma:

```
Lookup('Category', 'ProductID', ProductID, 'ProductList')
```

A tabela **ProductList** é carregada primeiro.

A **Lookup()** função é usada para construir a tabela **OrderData**. Especifica o terceiro argumento como **ProductID**. Este é o campo em que o valor deve ser pesquisado no segundo argumento **'ProductID'** na **ProductList**, conforme indicado pelas aspas simples.

A função retorna o valor para '**Category**' (na tabela **ProductList**), carregada como **CategoryID**.

O comando **drop** exclui a tabela **ProductList** do modelo de dados, porque ela não é necessária, o que deixa a tabela **OrderData** resultante.

A função Lookup() é flexível e pode acessar qualquer tabela previamente carregada. Contudo, é lenta em comparação com a função Applymap().

Consulte também:

ApplyMap (page 1331)

NoOfRows - função de gráfico

NoOfRows() retorna o número de linhas no atual segmento de coluna em uma tabela. Para gráficos de bitmap, **NoOfRows()** retorna o número de linhas no equivalente de tabela estática do gráfico.

Se a tabela ou o equivalente de tabela tiver várias dimensões verticais, o segmento de coluna atual incluirá somente linhas com os mesmos valores que a linha atual em todas as colunas de dimensão, exceto na coluna que mostrar a última dimensão na ordem de classificação entre os campos.

A classificação por valores y em gráficos ou por colunas de expressão em tabelas não é permitida quando essa função de gráfico é usada em qualquer uma das expressões do gráfico. Essas alternativas de classificação estão, portanto, automaticamente desabilitadas. Quando você usar essa função de gráfico em uma visualização ou tabela, a classificação da visualização será revertida para a entrada classificada dessa função.

Sintaxe:

```
NoOfRows([TOTAL])
```

Tipo de dados de retorno: inteiro

Argumentos:

Argumentos

Argumento	Descrição
TOTAL	Se a tabela for unidimensional ou se o qualificador **TOTAL** for usado como argumento, o segmento de coluna atual será sempre igual à coluna inteira.

Exemplo: Expressão de gráfico usando NoOfRows

Exemplo - expressão de gráfico

Script de carregamento

Carregue os seguintes dados como um carregamento inline no editor de carregamento de dados para criar os exemplos de expressão de gráfico abaixo.

```
        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 '|');
```

Expressão de gráfico

Crie uma visualização de tabela em uma pasta do Qlik Sense com **Region** e **SubRegion** como dimensões. Adicione `RowNo( )`, `NoOfRows()` e `NoOfRows(Total)` como medidas.

Resultado

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

Region	SubRegion	RowNo()	NoOfRows()	NoOfRows (Total)
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

Explicação

Neste exemplo, a ordem de classificação é pela primeira dimensão: Region. Como resultado, cada segmento de coluna é composto por um grupo de regiões que possuem o mesmo valor, por exemplo, África.

A coluna **RowNo()** mostra os números de linha para cada segmento de coluna, por exemplo, há duas linhas para a região África. Então, a numeração de coluna começa novamente em 1 para o próximo segmento de coluna, que é Americas.

A coluna **NoOfRows()** conta o número de linhas em cada segmento de coluna, por exemplo, Europa tem três linhas no segmento de coluna.

A coluna **NoOfRows(Total)** desconsidera as dimensões por causa do argumento TOTAL para NoOfRows() e conta as linhas da tabela.

Se a tabela fosse classificada pela segunda dimensão, SubRegion, os segmentos de coluna seriam baseados nessa dimensão, de modo que a numeração das linhas mudaria para cada SubRegion.

Consulte também:

RowNo - função de gráfico (page 594)

Peek

Peek() retorna o valor de um campo em uma tabela para uma linha que já foi carregada. O número da linha pode ser especificado, assim como a tabela. Se nenhum número de linha for especificado, o último registro carregado anteriormente será usado.

A função peek() é mais frequentemente usada para encontrar os limites relevantes em uma tabela carregada anteriormente, ou seja, o primeiro valor ou o último valor de um campo específico. Na maioria dos casos, esse valor é armazenado em uma variável para uso posterior, por exemplo, como uma condição em um loop do-while.

Sintaxe:

```
Peek (

field_name
```

```
[, row_no[, table_name ] ])
```

Tipo de dados de retorno: dual

Argumentos:

Argumentos

Argumento	Descrição
field_name	Nome do campo para o qual o valor de retorno é exigido.O valor de entrada deve ser fornecido como uma string (por exemplo, literais entre aspas).
row_no	A linha na tabela que especifica o campo obrigatório. Pode ser uma expressão, mas deve solucionar-se em um número inteiro. 0 indica o primeiro registro; 1, o segundo; e assim por diante. Números negativos indicam a ordem no final da tabela. -1 representa o último registro lido. Se nenhuma **row_no** for definida, assume-se -1.
table_name	Um rótulo de tabela sem os dois-pontos finais. Se nenhum **table_name**for definido, a tabela atual será assumida. Se usado fora do comando **LOAD** ou em referência a outra tabela, o **table_name** deve ser incluído.

Limitações:

A função só pode retornar valores de registros já carregados. Isso significa que, no primeiro registro de uma tabela, uma chamada usando -1 como row_no retornará NULL.

Exemplos e resultados:

Exemplo 1

Adicione o script de exemplo ao seu aplicativo e execute-o. Para ver o resultado, adicione os campos listados na coluna de resultados a uma pasta no seu aplicativo.

```
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;
```

Tabela resultante

Código do funcionário	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, pois `Peek('EmployeeCode',0, 'EmployeeDates')` retorna o primeiro valor de EmployeeCode na tabela EmployeeDates.

LastCode = 106, pois `Peek('EmployeeCode',-1, 'EmployeeDates')` retorna o último valor de EmployeeCode na tabela EmployeeDates.

Substituindo o valor do argumento, **row_no** retorna os valores de outras linhas na tabela, conforme a seguir:

O `Peek('EmployeeCode',2, 'EmployeeDates')` retorna o terceiro valor, 103, na tabela como FirstCode.

No entanto, observe que, sem especificar a tabela como **table_name** do terceiro argumento nesses exemplos, a função faz referência à tabela atual (neste caso, interna).

Exemplo 2

Se quiser acessar os dados mais abaixo em uma tabela, precisará fazer isso em duas etapas: primeiro, carregue a tabela inteira em uma tabela temporária e, em seguida, classifique-a novamente ao usar **Peek()**.

Adicione o script de exemplo ao seu aplicativo e execute-o. Para ver o resultado, adicione os campos listados na coluna de resultados a uma pasta no seu aplicativo.

```
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.

Tabela resultante

ID	Lista	Valor
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

O comando **IF()** é formado a partir da tabela temporária T1.
`Peek('ID')` referencia o campo ID na linha anterior da tabela atual T2.
`Peek('List')` referencia o campo List na linha prévia da tabela T2, atualmente sendo criada enquanto a expressão é avaliada.

A instrução é avaliada da seguinte maneira:
Se o valor atual de ID for o mesmo que o valor anterior de ID, escreva o valor de Peek('List') concatenado com o valor atual de Value. Senão, escreva somente o valor atual de Value.

Se Peek('List') já contém um resultado relacionado, o novo resultado de Peek('List') será relacionado a ele.

> *Observe a cláusula **Order by**. Ela especifica como a tabela é organizada (por ID em ordem crescente). Sem ela, a função Peek() usará qualquer ordenação arbitrária que a tabela interna possua, que pode acarretar em resultados imprevisíveis.*

Exemplo 3

Adicione o script de exemplo ao seu aplicativo e execute-o. Para ver o resultado, adicione os campos listados na coluna de resultados a uma pasta no seu aplicativo.

```
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];
```

Tabela resultante

Amount	AmountMonthBefore	Month
1	4	2022-06
2	-	2022-01
3	2	2022-02
4	9	2022-05
7	3	2022-03
9	7	2022-04

O campo AmountMonthBefore conterá o valor do mês anterior.

Aqui, os parâmetros row_no e table_name são omitidos e, portanto, os valores padrão são usados. Neste exemplo, as três chamadas de função a seguir são equivalentes:

- Peek(Amount)
- Peek(Amount,-1)
- Peek(Amount,-1,'Amounts')

Usar -1 como row_no significa que o valor da linha anterior será usado. Ao substituir esse valor, os valores de outras linhas na tabela podem ser obtidos:

Peek(Amount,2) retorna o terceiro valor na tabela: 7.

Exemplo 4

Os dados precisam ser classificados corretamente para obter os resultados corretos, mas, infelizmente, esse nem sempre é o caso. Além disso, a função Peek() não pode ser usada para referenciar dados que ainda não foram carregados. Usando tabelas temporárias e executando várias passagens pelos dados, esses problemas podem ser evitados.

Adicione o script de exemplo ao seu aplicativo e execute-o. Para ver o resultado, adicione os campos listados na coluna de resultados a uma pasta no seu aplicativo.

```
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;
```

Explicação

A tabela inicial é classificada de acordo com o mês, o que significa que a função peek() em muitos casos retornaria o valor do produto errado. Portanto, essa tabela precisa ser reclassificada. Isso é feito executando uma segunda passagem pelos dados, criando uma nova tabela tmp2Amounts. Observe a cláusula Order by. Ela ordena os registros primeiro por produto e depois por mês, em ordem crescente.

A função If() é necessária, pois AmountMonthBefore só deve ser calculado se a linha anterior contiver os dados para o mesmo produto, mas para o mês anterior. Ao comparar o produto da linha atual com o produto da linha anterior, essa condição pode ser validada.

Quando a segunda tabela é criada, a primeira tabela tmp1Amounts é eliminada usando um comando Drop Table.

Por fim, é feita uma terceira passagem pelos dados, mas agora com os meses classificados em ordem inversa. Dessa forma, AmountMonthAfter também pode ser calculado.

Cláusulas Order by especificam como a tabela é ordenada. Sem isso, a função Peek() usará qualquer ordenação arbitrária que a tabela interna tenha, o que pode levar a resultados imprevisíveis.

Resultado

Tabela resultante

Month	Product	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	-

Exemplo 5

Adicione o script de exemplo ao seu aplicativo e execute-o. Para ver o resultado, adicione os campos listados na coluna de resultados a uma pasta no seu aplicativo.

```
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;
```

Resultado

Tabela resultante

Trimestre	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

Explicação

A instrução de LOAD **Load *, rangesum(SumVal,peek('AccSumVal')) as AccSumVal** inclui uma chamada recursiva em que os valores anteriores são adicionados ao valor atual. Essa operação é utilizada para calcular um acúmulo de valores no script.

Consulte também:

Previous

Previous() encontra o valor da expressão **expr** usando dados do registro de entrada anterior que não foi descartado devido a uma cláusula **where**. No primeiro registro de uma tabela interna, a função retornará NULL.

Sintaxe:

```
Previous(expr)
```

Tipo de dados de retorno: dual

Argumentos:

Argumentos

Argumento	Descrição
expr	A expressão ou campo que contém os dados a serem medidos. A expressão pode conter funções **previous()** aninhadas para acessar os registros anteriores. Os dados são lidos diretamente da fonte de entrada, o que permite fazer referência também aos campos não carregados no Qlik Sense, isto é, mesmo que não tenham sido armazenados em seu banco de dados associado.

Limitações:

No primeiro registro de uma tabela interna, a função retorna NULL.

Exemplo:

Insira o seguinte em seu script de carregamento:

```
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 '|');
```

Ao usar a função **Previous()** no comando **Load**, é possível comparar o valor atual de Sales com o valor precedente e usar o resultado em um terceiro campo Increase.

Tabela resultante

Mês	Vendas.	Aumentar
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 - função de gráfico

Top() avalia uma expressão na primeira linha (superior) de um segmento de coluna em uma tabela. A linha para a qual ela é calculada depende do valor de **offset**; se estiver presente, o padrão será a linha superior. Para gráficos que não sejam tabelas, a avaliação **Top()** é feita na primeira linha da coluna atual, no equivalente de tabela estática do gráfico.

Sintaxe:

```
Top([TOTAL] expr [ , offset [,count ]])
```

Tipo de dados de retorno: dual

Argumentos:

Argumentos

Argumento	Descrição
expr	A expressão ou campo que contém os dados a serem medidos.

Argumento	Descrição
offset	Especificar um **offset** de n, maior que 1 move a avaliação da expressão para as linhas n abaixo da linha superior. Especificando um número de compensação negativa faz a função **Top** funcionar como a função **Bottom** com o número de compensação positiva correspondente.
count	Ao especificar um terceiro parâmetro **count** maior que 1, a função retornará uma série de valores **count**, um para cada **count** últimas linhas do segmento da coluna atual. Dessa forma, a função pode ser usada como um argumento para qualquer uma das funções de intervalo especiais. *Funções de intervalo (page 1338)*
TOTAL	Se a tabela for unidimensional ou se o qualificador **TOTAL** for usado como argumento, o segmento de coluna atual será sempre igual à coluna inteira.

Um segmento de coluna é definido como um subconjunto consecutivo de células que contêm os mesmos valores para as dimensões na ordem de classificação atual. As funções de inter-registro de gráficos são computadas no segmento de coluna, excluindo a dimensão mais à direita no gráfico de tabela simples equivalente. Se houver apenas uma dimensão no gráfico ou se o qualificador TOTAL for especificado, a expressão avalia a tabela completa.

Se a tabela ou o equivalente de tabela tiver várias dimensões verticais, o segmento de coluna atual incluirá somente linhas com os mesmos valores que a linha atual em todas as colunas de dimensão, exceto na coluna que mostrar a última dimensão na ordem de classificação entre os campos.

Limitações:

- As chamadas recursivas retornarão NULL.
- A classificação por valores y em gráficos ou por colunas de expressão em tabelas não é permitida quando essa função de gráfico é usada em qualquer uma das expressões do gráfico. Essas alternativas de classificação estão, portanto, automaticamente desabilitadas. Quando você usar essa função de gráfico em uma visualização ou tabela, a classificação da visualização será revertida para a entrada classificada dessa função.

Exemplos e resultados:

Exemplo: 1

Na captura de tela da tabela mostrada neste exemplo, a visualização de tabela é criada da dimensão **Customer** e das medições `Sum(Sales)` e `Top(Sum(Sales))`.

A coluna **Top(Sum(Sales))** retorna 587 para todas as linhas, porque esse é o valor da primeira linha: **Astrida**.

A tabela também mostra medições mais complexas: uma criada a partir de `Sum(Sales)+Top(Sum(Sales))` e uma rotulada como **Top offset 3**, que é criada usando a expressão `Sum(Sales)+Top(Sum(Sales), 3)` e tem o argumento **offset** definido como 3. Ela acrescenta o valor **Sum(Sales)** da linha atual ao valor da terceira linha a partir da linha superior, isto é, a linha atual mais o valor de **Canutility**.

Exemplo 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

Exemplo: 2

Nas capturas de telas das tabelas mostradas neste exemplo, mais dimensões foram adicionadas às visualizações: **Month** e **Product**. Para gráficos com mais de uma dimensão, os resultados de expressões que contêm as funções **Above**, **Below**, **Top** e **Bottom** dependem da ordem em que as dimensões da coluna são classificadas pelo Qlik Sense. O Qlik Sense avalia as funções com base nos segmentos de coluna que resultam da dimensão que é classificada por último. A ordem de classificação da coluna é controlada no painel de propriedades em **Classificação** e não é necessariamente a ordem em que as colunas aparecem na tabela.

Primeira tabela para o Exemplo 2. O valor de Top para a medida de First value baseada em 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

Segunda tabela para o Exemplo 2. O valor de Top para a medida de First value baseada em Product (AA para 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

Consulte o exemplo: 2 na função **Above** para obter detalhes adicionais.

Exemplo 3

Exemplo: 3	**Resultado**
A função **Top** pode ser usada como entrada para as funções range. Por exemplo: `RangeAvg (Top(Sum (Sales),1,3))`.	Nos argumentos para a função **Top()**, offset é definido como 1 e count é definido como 3. A função encontra os resultados da expressão **Sum (Sales)** em três linhas, com a linha acima da linha inferior no segmento de coluna (porque offset=1) e as duas linhas acima (onde há uma linha). Estes três valores são usados como entrada para a função RangeAvg(), que encontra a média de valores na variação de números fornecidos. Uma tabela com **Customer** como dimensão dá os seguintes resultados para a expressão 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 '|');
```

Consulte também:

- *Bottom - função de gráfico (page 1290)*
- *Above - função de gráfico (page 1282)*
- *Sum - função de gráfico (page 349)*
- *RangeAvg (page 1341)*
- *Funções de intervalo (page 1338)*

SecondaryDimensionality - função de gráfico

SecondaryDimensionality() retorna o número de linhas da tabela dinâmica da dimensão que têm conteúdo não-agregado, isto é, não contêm somas parciais ou agregações contraídas. Essa função é a equivalente da função **dimensionality()** para dimensões horizontais de tabela dinâmica.

Sintaxe:

```
SecondaryDimensionality( )
```

Tipo de dados de retorno: inteiro

Limitações:

- A menos que usada em tabelas dinâmicas, a função **SecondaryDimensionality** sempre retorna 0.
- A classificação por valores y em gráficos ou por colunas de expressão em tabelas não é permitida quando essa função de gráfico é usada em qualquer uma das expressões do gráfico. Essas alternativas de classificação estão, portanto, automaticamente desabilitadas. Quando você usar essa função de gráfico em uma visualização ou tabela, a classificação da visualização será revertida para a entrada classificada dessa função.

After - função de gráfico

After() retorna o valor de uma expressão avaliada com os valores de dimensão de uma tabela dinâmica como aparecerem na coluna depois da atual, em um segmento de linha da tabela dinâmica.

Sintaxe:

```
after([TOTAL] expr [, offset [, count ]])
```

A classificação por valores y em gráficos ou por colunas de expressão em tabelas não é permitida quando essa função de gráfico é usada em qualquer uma das expressões do gráfico. Essas alternativas de classificação estão, portanto, automaticamente desabilitadas. Quando você usar essa função de gráfico em uma visualização ou tabela, a classificação da visualização será revertida para a entrada classificada dessa função.

Esta função retorna NULL em todos os tipos de gráfico, exceto em tabelas dinâmicas.

Argumentos:

Argumento	Descrição
expr	A expressão ou campo que contém os dados a serem medidos.
offset	Especificar um **offset** n, maior que 1 move a avaliação da expressão para n linhas para a direita da linha atual. A especificação de um deslocamento igual a 0 avaliará a expressão na linha atual. Especificando um número de compensação negativa faz a função **After** funcionar como a função **Before** com o número de compensação positiva correspondente.
count	Ao especificar um terceiro parâmetro **count** maior que 1, a função retornará uma série de valores **count**, um para cada linhas da tabela, contado para a direita a partir da célula original.
TOTAL	Se a tabela for unidimensional ou se o qualificador **TOTAL** for usado como argumento, o segmento de coluna atual será sempre igual à coluna inteira.

Na última coluna de um segmento de linha, NULL um valor será retornado, já que não há uma coluna depois dela.

Se a tabela dinâmica tiver várias dimensões horizontais, o segmento de linha atual incluirá somente colunas com os mesmos valores que a coluna atual em todas as linhas de dimensão, exceto na linha que mostrar a última dimensão horizontal na ordem de classificação entre os campos. A ordem de classificação entre os campos para as dimensões horizontais das tabelas dinâmicas é definida simplesmente pela ordem das dimensões de cima para baixo.

Exemplo:

```
after( sum( Sales ))
after( sum( Sales ), 2 )
after( total sum( Sales ))
```

`rangeavg (after(sum(x),1,3))` retorna uma média dos três resultados da função **sum(x)** avaliada nas três colunas imediatamente à direita da atual.

Before - função de gráfico

Before() retorna o valor de uma expressão avaliada com os valores de dimensão de uma tabela dinâmica como aparecerem na coluna antes da atual, em um segmento de linha da tabela dinâmica.

Sintaxe:

```
before([TOTAL] expr [, offset [, count]])
```

Esta função retorna NULL em todos os tipos de gráfico, exceto em tabelas dinâmicas.

A classificação por valores y em gráficos ou por colunas de expressão em tabelas não é permitida quando essa função de gráfico é usada em qualquer uma das expressões do gráfico. Essas alternativas de classificação estão, portanto, automaticamente desabilitadas. Quando você usar essa função de gráfico em uma visualização ou tabela, a classificação da visualização será revertida para a entrada classificada dessa função.

Argumentos:

Argumentos

Argumento	Descrição
expr	A expressão ou campo que contém os dados a serem medidos.
offset	Especificar um **offset** n, maior que 1 move a avaliação da expressão para n linhas para a esquerda da linha atual. A especificação de um deslocamento igual a 0 avaliará a expressão na linha atual. Especificando um número de compensação negativa faz a função **Before** funcionar como a função **After** com o número de compensação positiva correspondente.
count	Ao especificar um terceiro parâmetro **count** maior que 1, a função retornará uma série de valores , um para cada **count** linhas da tabela, contado para a esquerda a partir da célula original.
TOTAL	Se a tabela for unidimensional ou se o qualificador **TOTAL** for usado como argumento, o segmento de coluna atual será sempre igual à coluna inteira.

Na primeira coluna de um segmento de linha, NULL um valor será retornado, já que não há uma coluna antes dele.

Se a tabela dinâmica tiver várias dimensões horizontais, o segmento de linha atual incluirá somente colunas com os mesmos valores que a coluna atual em todas as linhas de dimensão, exceto na linha que mostrar a última dimensão horizontal na ordem de classificação entre os campos.A ordem de classificação entre os campos para as dimensões horizontais das tabelas dinâmicas é definida simplesmente pela ordem das dimensões de cima para baixo.

Exemplos:

```
before( sum( Sales ))
before( sum( Sales ), 2 )
before( total sum( Sales ))
```

`rangeavg (before(sum(x),1,3))` retorna uma média dos três resultados da função **sum(x)** avaliada nas três colunas imediatamente à esquerda da atual.

First - função de gráfico

First() retorna o valor de uma expressão avaliada com os valores de dimensão de uma tabela dinâmica como aparecerem na primeira coluna do segmento de linha da tabela dinâmica. Esta função retorna NULL em todos os tipos de gráficos, exceto em tabelas dinâmicas.

A classificação por valores y em gráficos ou por colunas de expressão em tabelas não é permitida quando essa função de gráfico é usada em qualquer uma das expressões do gráfico. Essas alternativas de classificação estão, portanto, automaticamente desabilitadas. Quando você usar essa função de gráfico em uma visualização ou tabela, a classificação da visualização será revertida para a entrada classificada dessa função.

Sintaxe:

```
first([TOTAL] expr [, offset [, count]])
```

Argumentos:

Argumentos

Argumento	Descrição
expression	A expressão ou campo que contém os dados a serem medidos.
offset	Especificar um **offset** n, maior que 1 move a avaliação da expressão para n linhas para a direita da linha atual. A especificação de um deslocamento igual a 0 avaliará a expressão na linha atual. Especificando um número de compensação negativa faz a função **First** funcionar como a função **Last** com o número de compensação positiva correspondente.
count	Ao especificar um terceiro parâmetro **count** maior que 1, a função retornará uma série de valores **count**, um para cada linhas da tabela, contado para a direita a partir da célula original.
TOTAL	Se a tabela for unidimensional ou se o qualificador **TOTAL** for usado como argumento, o segmento de coluna atual será sempre igual à coluna inteira.

Se a tabela dinâmica tiver várias dimensões horizontais, o segmento de linha atual incluirá somente colunas com os mesmos valores que a coluna atual em todas as linhas de dimensão, exceto na linha que mostrar a última dimensão horizontal na ordem de classificação entre os campos.A ordem de classificação entre os campos para as dimensões horizontais das tabelas dinâmicas é definida simplesmente pela ordem das dimensões de cima para baixo.

Exemplos:

```
first( sum( Sales ))
first( sum( Sales ), 2 )
first( total sum( Sales )
```

`rangeavg (first(sum(x),1,5))` retorna uma média dos resultados da função **sum(x)** avaliada nas cinco colunas mais à esquerda do segmento de linha atual.

Last - função de gráfico

Last() retorna o valor de uma expressão avaliada com os valores de dimensão de uma tabela dinâmica como aparecerem na última coluna do segmento de linha da tabela dinâmica. Esta função retorna NULL em todos os tipos de gráficos, exceto em tabelas dinâmicas.

A classificação por valores y em gráficos ou por colunas de expressão em tabelas não é permitida quando essa função de gráfico é usada em qualquer uma das expressões do gráfico. Essas alternativas de classificação estão, portanto, automaticamente desabilitadas. Quando você usar essa função de gráfico em uma visualização ou tabela, a classificação da visualização será revertida para a entrada classificada dessa função.

Sintaxe:

```
last([TOTAL] expr [, offset [, count]])
```

Argumentos:

Argumentos

Argumento	Descrição
expr	A expressão ou campo que contém os dados a serem medidos.
offset	Especificar um **offset** n, maior que 1 move a avaliação da expressão para n linhas para a esquerda da linha atual. A especificação de um deslocamento igual a 0 avaliará a expressão na linha atual. Especificando um número de compensação negativa faz a função **First** funcionar como a função **Last** com o número de compensação positiva correspondente.
count	Ao especificar um terceiro parâmetro **count** maior que 1, a função retornará uma série de valores , um para cada **count** linhas da tabela, contado para a esquerda a partir da célula original.
TOTAL	Se a tabela for unidimensional ou se o qualificador **TOTAL** for usado como argumento, o segmento de coluna atual será sempre igual à coluna inteira.

Se a tabela dinâmica tiver várias dimensões horizontais, o segmento de linha atual incluirá somente colunas com os mesmos valores que a coluna atual em todas as linhas de dimensão, exceto na linha que mostrar a última dimensão horizontal na ordem de classificação entre os campos.A ordem de classificação entre os campos para as dimensões horizontais das tabelas dinâmicas é definida simplesmente pela ordem das dimensões de cima para baixo.

Exemplo:

```
last( sum( Sales ))
last( sum( Sales ), 2 )
last( total sum( Sales )
```

`rangeavg (last(sum(x),1,5))` retorna uma média de resultados da função **sum(x)** avaliada nas cinco colunas mais à direita do atual segmento de linha.

ColumnNo - função de gráfico

ColumnNo() retorna o número da coluna atual dentro do segmento da linha atual na tabela dinâmica. A primeira coluna é a de número 1.

Sintaxe:

```
ColumnNo([total])
```

Argumentos:

Argumentos

Argumento	Descrição
TOTAL	Se a tabela for unidimensional ou se o qualificador **TOTAL** for usado como argumento, o segmento de coluna atual será sempre igual à coluna inteira.

Se a tabela dinâmica tiver várias dimensões horizontais, o segmento de linha atual incluirá somente colunas com os mesmos valores que a coluna atual em todas as linhas de dimensão, exceto na linha que mostrar a última dimensão horizontal na ordem de classificação entre os campos.A ordem de classificação entre os campos para as dimensões horizontais das tabelas dinâmicas é definida simplesmente pela ordem das dimensões de cima para baixo.

A classificação por valores y em gráficos ou por colunas de expressão em tabelas não é permitida quando essa função de gráfico é usada em qualquer uma das expressões do gráfico. Essas alternativas de classificação estão, portanto, automaticamente desabilitadas. Quando você usar essa função de gráfico em uma visualização ou tabela, a classificação da visualização será revertida para a entrada classificada dessa função.

Exemplo:

```
if( ColumnNo( )=1, 0, sum( Sales ) / before( sum( Sales )))
```

NoOfColumns - função de gráfico

NoOfColumns() retorna o número de colunas no atual segmento da linha em uma tabela dinâmica.

A classificação por valores y em gráficos ou por colunas de expressão em tabelas não é permitida quando essa função de gráfico é usada em qualquer uma das expressões do gráfico. Essas alternativas de classificação estão, portanto, automaticamente desabilitadas. Quando você usar essa função de gráfico em uma visualização ou tabela, a classificação da visualização será revertida para a entrada classificada dessa função.

Sintaxe:

```
NoOfColumns([total])
```

Argumentos:

Argumentos

Argumento	Descrição
TOTAL	Se a tabela for unidimensional ou se o qualificador **TOTAL** for usado como argumento, o segmento de coluna atual será sempre igual à coluna inteira.

Se a tabela dinâmica tiver várias dimensões horizontais, o segmento de linha atual incluirá somente colunas com os mesmos valores que a coluna atual em todas as linhas de dimensão, exceto na linha que mostrar a última dimensão na ordem de classificação entre os campos.A ordem de classificação entre os campos para as dimensões horizontais das tabelas dinâmicas é definida simplesmente pela ordem das dimensões de cima para baixo.

Exemplo:

```
if( ColumnNo( )=NoOfColumns( ), 0, after( sum( Sales )))
```

5.17 Funções lógicas

Esta seção descreve as funções de manipulação de operações lógicas. Todas as funções podem ser usadas no script de carregamento de dados e em expressões de gráfico.

IsNum

Retornará -1 (True) se a expressão for interpretada como um número; caso contrário, retornará 0 (False).

```
IsNum( expr )
```

IsText

Retornará -1 (True) se a expressão tiver uma representação de texto; caso contrário, retornará 0 (False).

```
IsText( expr )
```

***IsNum** e **IsText** retornarão 0 se a expressão for NULL.*

Exemplo:

O exemplo a seguir carrega uma tabela inline com uma combinação de texto e valores numéricos e adiciona dois campos para conferir se o valor é um valor numérico e, respectivamente, um valor de texto.

```
Load *, IsNum(Value), IsText(Value)
Inline [
Value
23
Green
Blue
12
33Red];
```

A tabela resultante tem a seguinte aparência:

Resulting table

Value	IsNum(Value)	IsText(Value)
23	-1	0
Green	0	-1
Blue	0	-1
12	-1	0
33Red	0	-1

5.18 Funções de mapeamento

Esta seção descreve as funções de manipulação das tabelas de mapeamento. A tabela de mapeamento pode ser usada para substituir valores de campos ou nomes de campos durante a execução do script.

As funções de mapeamento podem ser usadas apenas no script de carregamento de dados.

Visão geral das funções de mapeamento

Cada função é descrita adicionalmente após a visão geral. Você também pode clicar no nome da função na sintaxe para acessar imediatamente os detalhes dessa função específica.

ApplyMap

A função de script **ApplyMap** é usada para mapear a saída de uma expressão para uma tabela de mapeamento carregada anteriormente.

```
ApplyMap ('mapname', expr [ , defaultexpr ] )
```

MapSubstring

A função de script **MapSubstring** é usada para mapear partes de qualquer expressão para uma tabela de mapeamento carregada anteriormente. O mapeamento diferencia maiúsculas de minúsculas e não iterativo, e as substrings são mapeadas da esquerda para a direita.

```
MapSubstring ('mapname', expr)
```

ApplyMap

A função de script **ApplyMap** é usada para mapear a saída de uma expressão para uma tabela de mapeamento carregada anteriormente.

Sintaxe:

```
ApplyMap('map_name', expression [ , default_mapping ] )
```

Tipo de dados de retorno: dual

Argumentos:

Argumentos

Argumento	Descrição
map_name	O nome de uma tabela de mapeamento que foi criada anteriormente com a declaração **mapping load** ou **mapping select**. O nome deve estar entre aspas simples. *Se você usar essa função em uma variável de macro expandida e fizer referência a uma tabela de mapeamento que não existe, a chamada de função falhará, e um campo não será criado.*
expression	A expressão cujo resultado deve ser mapeado.
default_ mapping	Se indicado, esse valor será usado como um valor padrão se a tabela de mapeamento não contiver nenhum valor correspondente a expression. Se nada for indicado, o valor de expression será retornado como está.

O campo de saída de ApplyMap não deve ter o mesmo nome que um dos campos de entrada. Isso pode causar resultados inesperados. Exemplo para não usar: `ApplyMap('Map', A) as A`.

Exemplo:

Neste exemplo, carregamos uma lista de vendedores com um código de país representando seu país de residência. A tabela é usada para mapear o código de um país a um país para substituir o código pelo nome do país. Somente três países estão definidos na tabela de mapeamento, outros códigos de países estão mapeados para 'Rest of the world'.

```
// 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';
```

A tabela resultante (Salespersons) se parece com isto:

Resulting table

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

MapSubstring

A função de script **MapSubstring** é usada para mapear partes de qualquer expressão para uma tabela de mapeamento carregada anteriormente. O mapeamento diferencia maiúsculas de minúsculas e não iterativo, e as substrings são mapeadas da esquerda para a direita.

Sintaxe:

```
MapSubstring('map_name', expression)
```

Tipo de dados de retorno: caractere

Argumentos:

Argumentos

Argumento	Descrição
map_name	O nome de uma tabela de mapeamento lida anteriormente em um comando **mapping load** ou **mapping select**. O nome deve estar entre aspas simples retas. *Se você usar essa função em uma variável de macro expandida e fizer referência a uma tabela de mapeamento que não existe, a chamada de função falhará, e um campo não será criado.*
expression	É a expressão cujo resultado deve ser mapeado pelas substrings.

Exemplo:

Neste exemplo, carregamos uma lista de modelos de produtos. Cada modelo tem um conjunto de atributos descritos por um código composto. Usando a tabela de mapeamento com MapSubstring, podemos expandir os códigos de atributos a uma descrição.

```
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';
```

A tabela resultante tem a seguinte aparência:

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

5.19 Funções matemáticas

Esta seção descreve as funções de constantes matemáticas e valores boolianos. Essas funções não têm parâmetros, mas os parênteses ainda são necessários.

Todas as funções podem ser usadas no script de carregamento de dados e em expressões de gráfico.

e
A função retorna a base dos logaritmos naturais, **e**.(2,71828…)

```
e( )
```

false
A função retorna um valor dual com valor de texto 'False' e valor numérico 0, que pode ser usado como falso lógico em expressões.

```
false( )
```

pi
A função retorna o valor de π (3,14159…)

```
pi( )
```

rand
A função retorna um número aleatório entre 0 e 1. Isso pode ser usado para criar dados de amostra.

```
rand( )
```

Exemplo:

Esse exemplo de script cria uma tabela de 1000 registros com caracteres maiúsculos selecionados aleatoriamente, ou seja, caracteres no intervalo de 65 a 91 (65+26).

```
Load
     Chr( Floor(rand() * 26) + 65) as UCaseChar,
     RecNo() as ID
     Autogenerate 1000;
```

true
A função retorna um valor dual com valor de texto 'True' e valor numérico -1, que pode ser usado como verdadeiro lógico em expressões.

```
true( )
```

5.20 Funções NULL

Esta seção descreve as funções para retornar ou detectar valores NULL.

Todas as funções podem ser usadas no script de carregamento de dados e em expressões de gráfico.

Visão geral das funções NULL

Cada função é descrita adicionalmente após a visão geral. Você também pode clicar no nome da função na sintaxe para acessar imediatamente os detalhes dessa função específica.

EmptyIsNull
A função **EmptyIsNull** converte strings vazias em NULL. Portanto, ela retornará NULL se o parâmetro for uma string vazia. Caso contrário, retornará o parâmetro.

```
EmptyIsNull (expr )
```

IsNull
A função **IsNull** testa se o valor de uma expressão é NULL e, nesse caso, retorna -1 (True), do contrário 0 (False).

```
IsNull (expr )
```

Null
A função **Null** retorna um valor NULL.

```
NULL( )
```

EmptyIsNull

A função **EmptyIsNull** converte strings vazias em NULL. Portanto, ela retornará NULL se o parâmetro for uma string vazia. Caso contrário, retornará o parâmetro.

Sintaxe:

```
EmptyIsNull(exp )
```

Exemplos e resultados:

Exemplos de scripts

Exemplo	Resultado
`EmptyIsNull(AdditionalComments)`	Essa expressão retornará como nulo quaisquer valores de string vazios do *campo* AdditionalComments, em vez de strings vazias. Strings e números não vazios são retornados.
`EmptyIsNull(PurgeChar (PhoneNumber, ' -()'))`	Essa expressão removerá quaisquer traços, espaços e parênteses do campo *PhoneNumber*. Se não houver caracteres restantes, a função EmptyIsNull retornará a string vazia como nula. Um número de telefone vazio equivale a nenhum número de telefone.

IsNull

A função **IsNull** testa se o valor de uma expressão é NULL e, nesse caso, retorna -1 (True), do contrário 0 (False).

Sintaxe:

```
IsNull(expr )
```

*Uma string de comprimento zero não é considerada um NULL e fará **IsNull** retornar False.*

Exemplo: Script de carregamento de dados

Neste exemplo, carregamos uma tabela inline de quatro linhas, em que as primeiras três contém -,'NULL' ou nada na coluna Value. Convertemos esses valores em representações de valor NULL verdadeiro com o **LOAD** precedente intermediário, usando a função **Null**.

O primeiro **LOAD** precedente adiciona um campo que confere se o valor é NULL, usando a função **IsNull**.

```
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];
```

Esta é a tabela resultante. Na coluna ValueNullConv, os valoresNULL são representados por -.

Resulting table

ID	Value	ValueNullConv	IsItNull
0		-	T
1	NULL	-	T
2	-	-	T
3	Value	Value	F

NULL

A função **Null** retorna um valor NULL.

Sintaxe:

```
Null( )
```

Exemplo: Script de carregamento de dados

Neste exemplo, carregamos uma tabela inline de quatro linhas, em que as primeiras três contém -,'NULL' ou nada na coluna Value. O objetivo é converter valores em representações de valor NULL verdadeiro.

O **LOAD** precedente intermediário realiza a conversão usando a função **Null**.

O primeiro **LOAD** precedente adiciona um campo que confere se o valor é NULL, com o objetivo de ilustração neste exemplo.

```
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];
```

Esta é a tabela resultante. Na coluna ValueNullConv, os valoresNULL são representados por -.

Resulting table

ID	Value	ValueNullConv	IsItNull
0		-	T
1	NULL	-	T
2	-	-	T
3	Value	Value	F

5.21 Funções de intervalo

As funções de intervalo são funções que pegam um conjunto de valores e produzem um valor único como resultado. Todas as funções de intervalo podem ser usadas em scripts de carga de dados e em expressões de gráfico.

Por exemplo, em uma visualização, uma função de intervalo pode calcular um único valor a partir de um conjunto de inter-registro. No script de carga dos dados, uma função de intervalo pode calcular um único valor a partir de um conjunto de valores em uma tabela interna.

*As funções de intervalo substituem as seguintes funções numéricas gerais: **numsum**, **numavg**, **numcount**, **nummin** e **nummax**, que devem agora ser consideradas obsoletas.*

Funções básicas de intervalo

RangeMax
RangeMax() retorna os valores numéricos mais altos encontrados na expressão ou campo.

```
RangeMax (first_expr[, Expression])
```

RangeMaxString
RangeMaxString() retorna o último valor da ordem de classificação de texto encontrado na expressão ou campo.

```
RangeMaxString (first_expr[, Expression])
```

RangeMin
RangeMin() retorna os menores valores numéricos encontrados na expressão ou campo.

```
RangeMin (first_expr[, Expression])
```

RangeMinString
RangeMinString() retorna o primeiro valor na ordem de classificação de textos encontrado na expressão ou campo.

```
RangeMinString (first_expr[, Expression])
```

RangeMode
RangeMode() encontra o valor mais geralmente ocorrido (valor de modo) na expressão ou campo.

```
RangeMode (first_expr[, Expression])
```

RangeOnly
RangeOnly() é uma função dupla que retorna um valor se a expressão for avaliada como um valor único. Se não for esse o caso, **NULL** é retornado.

```
RangeOnly (first_expr[, Expression])
```

RangeSum

O **RangeSum()** retorna a soma de um intervalo de valores. Todos os valores não numéricos são tratados como 0.

```
RangeSum (first_expr[, Expression])
```

Funções de intervalo de contador

RangeCount

RangeCount() retorna o número de valores, de texto e numéricos, em uma expressão ou campo.

```
RangeCount (first_expr[, Expression])
```

RangeMissingCount

RangeMissingCount() retorna o número de valores não numéricos (incluindo NULL) na expressão ou campo.

```
RangeMissingCount (first_expr[, Expression])
```

RangeNullCount

RangeNullCount() encontra o número de valores NULL na expressão ou campo.

```
RangeNullCount (first_expr[, Expression])
```

RangeNumericCount

RangeNumericCount() encontra o número de valores numéricos em uma expressão ou campo.

```
RangeNumericCount (first_expr[, Expression])
```

RangeTextCount

RangeTextCount() retorna o número de valores de texto em uma expressão ou campo.

```
RangeTextCount (first_expr[, Expression])
```

Funções de intervalo estatístico

RangeAvg

RangeAvg() retorna a média de um intervalo. A entrada para a função pode ser um intervalo de valores ou uma expressão.

```
RangeAvg (first_expr[, Expression])
```

RangeCorrel

RangeCorrel() retorna o coeficiente da correlação para dois conjuntos de dados. O coeficiente da correlação é uma medida da relação entre os conjuntos de dados.

```
RangeCorrel (x_values , y_values[, Expression])
```

RangeFractile

RangeFractile() retorna o valor que corresponde ao enésimo **fractile** (quantil) de um intervalo de números.

```
RangeFractile (fractile, first_expr[ ,Expression])
```

RangeKurtosis

RangeKurtosis() retorna o valor que corresponde à curtose de um intervalo de números.

```
RangeKurtosis (first_expr[, Expression])
```

RangeSkew

RangeSkew() é uma função de gráfico numérica que retorna o valor correspondente à assimetria de um intervalo de números.

```
RangeSkew (first_expr[, Expression])
```

RangeStdev

RangeStdev() localiza o desvio padrão de um intervalo de números.

```
RangeStdev (expr1[, Expression])
```

Funções de intervalo financeiro

RangeIRR

RangeIRR() retorna a taxa de retorno interno de uma série de fluxos de caixa representada pelos valores de entrada.

```
RangeIRR (value[, value][, Expression])
```

RangeNPV

RangeNPV() retorna o valor líquido atual de um investimento com base em uma taxa de desconto e em uma série de pagamentos periódicos futuros (valores negativos) e receitas (valores positivos). O resultado apresenta um formato numérico padrão de **money**.

```
RangeNPV (discount_rate, value[, value][, Expression])
```

RangeXIRR

RangeXIRR() retorna a taxa interna de retorno (anual) para uma programação de fluxos de caixa que não é necessariamente periódica. Para calcular a taxa de retorno interno de uma série de fluxos de caixa periódicos, use a função **RangeIRR**.

```
RangeXIRR (values, dates[, Expression])
```

RangeXNPV

RangeXNPV() retorna o valor presente líquido para um cronograma de fluxos de caixa (não necessariamente periódicos) representados por números emparelhados nas expressões especificadas por **pmt** e **date**. Todos os pagamentos têm descontos baseados em um ano de 365 dias.

```
RangeXNPV (discount_rate, values, dates[, Expression])
```

Consulte também:

Funções interregistro (page 1278)

RangeAvg

RangeAvg() retorna a média de um intervalo. A entrada para a função pode ser um intervalo de valores ou uma expressão.

Sintaxe:

```
RangeAvg(first_expr[, Expression])
```

Tipo de dados de retorno: numérico

Argumentos:

Os argumentos dessa função podem conter funções interregistro, que por si só retornam um intervalo de valores.

Argumentos

Argumento	Descrição
first_expr	A expressão ou campo que contém os dados a serem medidos.
Expression	Expressões ou campos opcionais que contêm o intervalo de dados a ser medido.

Limitações:

Se não for encontrado nenhum valor numérico, será retornado NULL.

Exemplos e resultados:

Exemplos de scripts

Exemplos	Resultados
`RangeAvg (1,2,4)`	Retorna 2,33333333
`RangeAvg (1,'xyz')`	Retorna 1
`RangeAvg (null( ), 'abc')`	Retorna NULL

Exemplo:

Adicione o script de exemplo ao seu aplicativo e execute-o. Para ver o resultado, adicione os campos listados na coluna de resultados a uma pasta no seu aplicativo.

```
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
];
```

A tabela resultante mostra os valores retornados de MyRangeAvg para cada um dos registros na tabela.

Tabela resultante

RangeID	MyRangeAvg
1	7
2	4
3	6
4	12.666
5	6.333
6	5

Exemplo com a expressão:

```
RangeAvg (Above(MyField),0,3))
```

Retorna uma média variável do resultado do intervalo dos três valores de **MyField** calculado na linha atual e nas duas linhas acima da atual. Ao especificar o terceiro argumento como 3, a função **Above()** retorna três valores, em que há linhas suficientes acima, que são tomadas como entrada na função **RangeAvg()**.

Dados usados nos exemplos:

*Desative a classificação de **MyField** para garantir que o exemplo funcione como esperado.*

Dados de exemplo

MyField	RangeAvg (Above (MyField,0,3))	Comments
10	10	Com esta é a linha do topo, o intervalo consiste em apenas um valor.
2	6	Há apenas uma linha acima desta linha e, portanto, o intervalo é: 10,2.
8	6.6666666667	O equivalente a RangeAvg(10,2,8)
18	9.333333333	-
5	10. 333333333	-
9	10.6666666667	-

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
```

```
] ;
```

Consulte também:

- *Avg - função de gráfico (page 404)*
- *Count - função de gráfico (page 354)*

RangeCorrel

RangeCorrel() retorna o coeficiente da correlação para dois conjuntos de dados. O coeficiente da correlação é uma medida da relação entre os conjuntos de dados.

Sintaxe:

```
RangeCorrel(x_value , y_value[, Expression])
```

Tipo de dados de retorno: numérico

A série de dados deve ser inserida como pares (x,y). Por exemplo, para avaliar duas séries de dados, matriz 1 e matriz 2, onde a matriz 1 = 2,6,9 e matriz 2 = 3,8,4 você escreveria `RangeCorrel (2,3,6,8,9,4)` que retorna 0,269.

Argumentos:

Argumentos

Argumento	Descrição
x-value, y-value	Cada valor representa um único valor ou um intervalo de valores retornado por funções inter-registro com um terceiro parâmetro opcional. Cada valor ou intervalo de valores deve corresponder a um valor **x-value** ou a um intervalo de valores **y-values**.
Expression	Expressões ou campos opcionais que contêm o intervalo de dados a ser medido.

Limitações:

São necessários, pelo menos, dois pares de coordenadas para a função ser calculada.

Os valores de texto NULL e faltantes retornam NULL.

Exemplos e resultados:

Exemplos de funções

Exemplos	Resultados
`RangeCorrel (2,3,6,8,9,4,8,5)`	Retorna 0,2492. Essa função pode ser carregada no script ou adicionada em uma visualização no editor de expressões.

Exemplo:

Adicione o script de exemplo ao seu aplicativo e execute-o. Para ver o resultado, adicione os campos listados na coluna de resultados a uma pasta no seu aplicativo.

```
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 '|');
```

Em uma tabela com ID1 como uma dimensão e a medida RangeCorrel(x1,y1,x2,y2,x3,y3,x4,y4,x5,y5,x6,y6)), a função **RangeCorrel()** encontra o valor de **Correl** em um intervalo de seis pares de x,y, para cada um dos valores de ID1.

Tabela resultante

ID1	MyRangeCorrel
01	-0.9517
02	-0.5209
03	-0.5209
04	-0.1599

Exemplo:

```
XY:
LOAD recno() as RangeID, * Inline [
X|Y
2|3
6|8
9|4
8|5
](delimiter is '|');
```

Em uma tabela com RangeID como uma dimensão e a medida: RangeCorrel(Below(X,0,4,BelowY,0,4)), a função **RangeCorrel()** usa os resultados das funções **Below()**, que, devido ao terceiro argumento (count) definido como 4, produz um intervalo de quatro valores x-y da tabela carregada XY.

Tabela resultante

RangeID	MyRangeCorrel2
01	0.2492
02	-0.9959
03	-1.0000
04	-

O valor para RangeID 01 é o mesmo que inserido manualmente RangeCorrel(2,3,6,8,9,4,8,5). Para os outros valores de RangeID, as séries produzidas pela função Below() são: (6,8,9,4,8,5), (9,4,8,5) e (8,5), o último que produz um resultado nulo.

Consulte também:

Correl - função de gráfico (page 408)

RangeCount

RangeCount() retorna o número de valores, de texto e numéricos, em uma expressão ou campo.

Sintaxe:

```
RangeCount(first_expr[, Expression])
```

Tipo de dados de retorno: inteiro

Argumentos:

Os argumentos dessa função podem conter funções interregistro, que por si só retornam um intervalo de valores.

Argumentos

Argumento	Descrição
first_expr	A expressão ou campo que contém os dados a serem contados.
Expression	Expressões ou campos opcionais que contêm o intervalo de dados a ser contado.

Limitações:

Valores NULL não são contados.

Exemplos e resultados:

Exemplos de funções

Exemplos	Resultados
`RangeCount (1,2,4)`	Retorna 3
`RangeCount (2,'xyz')`	Retorna 2
`RangeCount (null( ))`	Retorna 0
`RangeCount (2,'xyz', null())`	Retorna 2

Exemplo:

Adicione o script de exemplo ao seu aplicativo e execute-o. Para ver o resultado, adicione os campos listados na coluna de resultados a uma pasta no seu aplicativo.

```
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
];
```

A tabela resultante mostra os valores retornados de MyRangeCount para cada um dos registros na tabela.

Tabela de resultados

RangeID	MyRangeCount
1	3
2	3
3	3
4	3
5	3
6	3

Exemplo com a expressão:

```
RangeCount (Above(MyField,1,3))
```

Retorna o número de valores contidos nos três resultados do **MyField**. Ao especificar o primeiro argumento da função **Above()** como 1 e o segundo argumento como 3, retorna os valores dos três primeiros campos acima da linha atual, onde há linhas suficientes, que são utilizadas como entrada para a função **RangeCount()**.

Dados usados nos exemplos:

Dados de exemplo

MyField	RangeCount(Above(MyField,1,3))
10	0
2	1
8	2
18	3
5	3
9	3

Dados usados nos exemplos:

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
] ;
```

Consulte também:

Count - função de gráfico (page 354)

RangeFractile

RangeFractile() retorna o valor que corresponde ao enésimo **fractile** (quantil) de um intervalo de números.

RangeFractile() utiliza interpolação linear entre as classificações mais próximas quando calcula a fração.

Sintaxe:

```
RangeFractile(fractile, first_expr[, Expression])
```

Tipo de dados de retorno: numérico

Argumentos:

Os argumentos dessa função podem conter funções interregistro, que por si só retornam um intervalo de valores.

Argumentos

Argumento	Descrição
fractile	Um número entre 0 e 1 correspondente ao fractil (quantil expressado como uma fração) que será calculado.
first_expr	A expressão ou campo que contém os dados a serem medidos.
Expression	Expressões ou campos opcionais que contêm o intervalo de dados a ser medido.

Exemplos e resultados:

Exemplos de funções

Exemplos	Resultados
`RangeFractile (0.24,1,2,4,6)`	Retorna 1,72
`RangeFractile(0.5,1,2,3,4,6)`	Retorna 3
`RangeFractile (0.5,1,2,5,6)`	Retorna 3,5

Exemplo:

Adicione o script de exemplo ao seu aplicativo e execute-o. Para ver o resultado, adicione os campos listados na coluna de resultados a uma pasta no seu aplicativo.

```
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
];
```

A tabela resultante mostra os valores retornados de MyRangeFrac para cada um dos registros na tabela.

Tabela resultante

RangeID	MyRangeFrac
1	6
2	3
3	8
4	11
5	5
6	4

Exemplo com a expressão:

```
RangeFractile (0.5, Above(Sum(MyField),0,3))
```

Neste exemplo, a função inter-registro **Above()** contém o opcional offset e count argumentos. Isso produz uma variedade de resultados que podem ser usados como entrada em qualquer uma das variações de funções. Neste caso, `Above(Sum(MyField),0,3)` retorna os valores de `MyField` para a linha atual e as duas linhas acima. Esses valores fornecem entrada para a função **RangeFractile()**. Então, para a linha inferior na tabela abaixo, isso é o equivalente a `RangeFractile(0.5, 3,4,6)`, ou seja, calcular a fração 0.5 para a série 3, 4 e 6. As duas primeiras filas na tabela abaixo, o número de valores na variação é reduzido de acordo, onde não existem linhas acima da linha atual. Resultados semelhantes são produzidos para outras funções inter-registro.

Dados de exemplo

MyField	RangeFractile(0.5, Above(Sum(MyField),0,3))
1	1
2	1.5
3	2
4	3
5	4
6	5

Dados usados nos exemplos:

```
RangeTab:
LOAD * INLINE [
MyField
1
2
3
4
5
6
] ;
```

Consulte também:

- *Above - função de gráfico (page 1282)*
- *Fractile - função de gráfico (page 412)*

RangeIRR

RangeIRR() retorna a taxa de retorno interno de uma série de fluxos de caixa representada pelos valores de entrada.

A taxa de retorno interno é a taxa de juros recebida em um investimento que consiste em pagamentos (valores negativos) e receita (valores positivos) que ocorrem em períodos regulares.

Essa função usa uma versão simplificada do método de Newton para calcular a taxa de retorno interna (TIR).

Sintaxe:

```
RangeIRR(value[, value][, Expression])
```

Tipo de dados de retorno: numérico

Argumentos

Argumento	Descrição
value	Um único valor ou um intervalo de valores retornado por uma função inter-registro com um terceiro parâmetro opcional. A função precisa de pelo menos um valor positivo e um valor negativo para ser calculada.
Expression	Expressões ou campos opcionais que contêm o intervalo de dados a ser medido.

Limitações:

Os valores de texto, os valores NULL e os valores ausentes são ignorados.

Tabela de exemplo

Exemplos	Resultados
`RangeIRR(-70000,12000,15000,18000,21000,26000)`	Retorna 0,0866

Adicione o script de exemplo ao seu aplicativo e execute-o. Para ver o resultado, adicione os campos listados na coluna de resultados a uma pasta no seu aplicativo.
`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 '|');`
	A tabela resultante mostra os valores retornados de RangeIRR para cada um dos registros na tabela.
	RangeID — RangeIRR
	1 — 0.0639
	2 — 0.8708
	3 — -
	4 — 5.8419
	5 — 0.9318
	6 — -0.2566

Consulte também:

Funções interregistro (page 1278)

RangeKurtosis

RangeKurtosis() retorna o valor que corresponde à curtose de um intervalo de números.

Sintaxe:

```
RangeKurtosis(first_expr[, Expression])
```

Tipo de dados de retorno: numérico

Argumentos:

Os argumentos dessa função podem conter funções interregistro, que por si só retornam um intervalo de valores.

Argumentos

Argumento	Descrição
first_expr	A expressão ou campo que contém os dados a serem medidos.
Expression	Expressões ou campos opcionais que contêm o intervalo de dados a ser medido.

Limitações:

Se não for encontrado nenhum valor numérico, será retornado NULL.

Exemplos e resultados:

Exemplos de funções

Exemplos	Resultados
`RangeKurtosis (1,2,4,7)`	Retorna -0,28571428571429

Consulte também:

Kurtosis - função de gráfico (page 420)

RangeMax

RangeMax() retorna os valores numéricos mais altos encontrados na expressão ou campo.

Sintaxe:

```
RangeMax(first_expr[, Expression])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
first_expr	A expressão ou campo que contém os dados a serem medidos.
Expression	Expressões ou campos opcionais que contêm o intervalo de dados a ser medido.

Limitações:

Se não for encontrado nenhum valor numérico, será retornado NULL.

Exemplos e resultados:

Exemplos de funções

Exemplos	Resultados
`RangeMax (1,2,4)`	Retorna 4
`RangeMax (1,'xyz')`	Retorna 1
`RangeMax (null( ), 'abc')`	Retorna NULL

Exemplo:

Adicione o script de exemplo ao seu aplicativo e execute-o. Para ver o resultado, adicione os campos listados na coluna de resultados a uma pasta no seu aplicativo.

```
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
];
```

A tabela resultante mostra os valores retornados de MyRangeMax para cada um dos registros na tabela.

Tabela resultante

RangeID	MyRangeMax
1	10
2	7
3	8
4	18
5	9
6	9

Exemplo com a expressão:

```
RangeMax (Above(MyField,0,3))
```

Retorna o valor máximo do intervalo dos três valores de **MyField** calculado na linha atual e nas duas linhas acima da atual. Ao especificar o terceiro argumento como 3, a função **Above()** retorna três valores, em que há linhas suficientes acima, que são tomadas como entrada na função **RangeMax()**.

Dados usados nos exemplos:

*Desative a classificação de **MyField** para garantir que o exemplo funcione como esperado.*

Dados de exemplo

MyField	RangeMax (Above(Sum(MyField),1,3))
10	10
2	10
8	10
18	18
5	18
9	18

Dados usados nos exemplos:

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
] ;
```

RangeMaxString

RangeMaxString() retorna o último valor da ordem de classificação de texto encontrado na expressão ou campo.

Sintaxe:

```
RangeMaxString(first_expr[, Expression])
```

Tipo de dados de retorno: caractere

Argumentos:

Os argumentos dessa função podem conter funções interregistro, que por si só retornam um intervalo de valores.

Argumentos

Argumento	Descrição
first_expr	A expressão ou campo que contém os dados a serem medidos.
Expression	Expressões ou campos opcionais que contêm o intervalo de dados a ser medido.

Exemplos e resultados:

Exemplos de funções

Exemplos	Resultados
`RangeMaxString (1,2,4)`	Retorna 4
`RangeMaxString ('xyz','abc')`	Retorna 'xyz'
`RangeMaxString (5,'abc')`	Retorna 'abc'
`RangeMaxString (null( ))`	Retorna NULL

Exemplo com a expressão:

```
RangeMaxString (Above(MaxString(MyField),0,3))
```

Retorna o último (ordem de classificação do texto) dos três resultados da função **MaxString(MyField)** avaliada na linha atual e duas linhas acima da atual.

Dados usados nos exemplos:

*Desative a classificação de **MyField** para garantir que o exemplo funcione como esperado.*

Dados de exemplo

MyField	RangeMaxString(Above(MaxString(MyField),0,3))
10	10
abc	abc
8	abc
def	def
xyz	xyz
9	xyz

Dados usados nos exemplos:

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
8
'def'
```

```
'xyz'
9
] ;
```

Consulte também:

- *MaxString - função de gráfico (page 545)*

RangeMin

RangeMin() retorna os menores valores numéricos encontrados na expressão ou campo.

Sintaxe:

```
RangeMin(first_expr[, Expression])
```

Tipo de dados de retorno: numérico

Argumentos:

Argumentos

Argumento	Descrição
first_expr	A expressão ou campo que contém os dados a serem medidos.
Expression	Expressões ou campos opcionais que contêm o intervalo de dados a ser medido.

Limitações:

Se não for encontrado nenhum valor numérico, será retornado NULL.

Exemplos e resultados:

Exemplos de funções

Exemplos	Resultados
`RangeMin (1,2,4)`	Retorna 1
`RangeMin (1,'xyz')`	Retorna 1
`RangeMin (null( ), 'abc')`	Retorna NULL

Exemplo:

Adicione o script de exemplo ao seu aplicativo e execute-o. Para ver o resultado, adicione os campos listados na coluna de resultados a uma pasta no seu aplicativo.

```
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
];
```

A tabela resultante mostra os valores retornados de MyRangeMin para cada um dos registros na tabela.

Tabela resultante

RangeID	MyRangeMin
1	5
2	2
3	2
4	9
5	5
6	2

Exemplo com a expressão:

```
RangeMin (Above(MyField,0,3)
```

Retorna o valor mínimo do intervalo dos três valores de **MyField** calculado na linha atual e nas duas linhas acima da atual. Ao especificar o terceiro argumento como 3, a função **Above()** retorna três valores, em que há linhas suficientes acima, que são tomadas como entrada na função **RangeMin()**.

Dados usados nos exemplos:

Dados de exemplo

MyField	RangeMin(Above(MyField,0,3))
10	10
2	2
8	2
18	2
5	5
9	5

Dados usados nos exemplos:

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
] ;
```

Consulte também:

Min - função de gráfico (page 341)

RangeMinString

RangeMinString() retorna o primeiro valor na ordem de classificação de textos encontrado na expressão ou campo.

Sintaxe:

```
RangeMinString(first_expr[, Expression])
```

Tipo de dados de retorno: caractere

Argumentos:

Os argumentos dessa função podem conter funções interregistro, que por si só retornam um intervalo de valores.

Argumentos

Argumento	Descrição
first_expr	A expressão ou campo que contém os dados a serem medidos.
Expression	Expressões ou campos opcionais que contêm o intervalo de dados a ser medido.

Exemplos e resultados:

Exemplos de funções

Exemplos	Resultados
`RangeMinString (1,2,4)`	Retorna 1
`RangeMinString ('xyz','abc')`	Retorna 'abc'
`RangeMinString (5,'abc')`	Retorna 5
`RangeMinString (null( ))`	Retorna NULL

Exemplo com a expressão:

```
RangeMinString (Above(MinString(MyField),0,3))
```

Retorna o primeiro (ordem de classificação do texto) dos três resultados da função **MinString(MyField)** avaliada na linha atual e duas linhas acima da atual.

Dados usados nos exemplos:

*Desative a classificação de **MyField** para garantir que o exemplo funcione como esperado.*

Dados de exemplo

MyField	RangeMinString(Above(MinString(MyField),0,3))
10	10
abc	10
8	8
def	8
xyz	8
9	9

Dados usados nos exemplos:

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
8
'def'
'xyz'
9
] ;
```

Consulte também:

MinString - função de gráfico (page 547)

RangeMissingCount

RangeMissingCount() retorna o número de valores não numéricos (incluindo NULL) na expressão ou campo.

Sintaxe:

```
RangeMissingCount(first_expr[, Expression])
```

Tipo de dados de retorno: inteiro

Argumentos:

Os argumentos dessa função podem conter funções interregistro, que por si só retornam um intervalo de valores.

Argumentos

Argumento	Descrição
first_expr	A expressão ou campo que contém os dados a serem contados.
Expression	Expressões ou campos opcionais que contêm o intervalo de dados a ser contado.

Exemplos e resultados:

Exemplos de funções

Exemplos	Resultados
`RangeMissingCount (1,2,4)`	Retorna 0
`RangeMissingCount (5,'abc')`	Retorna 1
`RangeMissingCount (null( ))`	Retorna 1

Exemplo com a expressão:

```
RangeMissingCount (Above(MinString(MyField),0,3))
```

Retorna o número de valores não numéricos nos três resultados da função **MinString(MyField)** avaliada na linha atual e duas linhas acima da atual.

*Desative a classificação de **MyField** para garantir que o exemplo funcione como esperado.*

Dados de exemplo

MyField	RangeMissingCount (Above(MinString (MyField),0,3))	Explanation
10	2	Retorna 2 porque não existem linhas acima desta linha, portanto 2 dos 3 valores estão ausentes.
abc	2	Retorna 2 porque existe apenas 1 linha acima da linha atual e a linha atual é não numérica ('abc').
8	1	Retorna 1 porque 1 das 3 linhas inclui um ('abc') não numérico.
def	2	Retorna 2 porque 2 das 3 linhas incluem valores não numéricos ('def' e 'abc').
xyz	2	Retorna 2 porque 2 das 3 linhas incluem valores não numéricos (' xyz' e 'def').
9	2	Retorna 2 porque 2 das 3 linhas incluem valores não numéricos (' xyz' e 'def').

Dados usados nos exemplos:

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
8
'def'
'xyz'
```

```
9
] ;
```

Consulte também:

MissingCount - função de gráfico (page 358)

RangeMode

RangeMode() encontra o valor mais geralmente ocorrido (valor de modo) na expressão ou campo.

Sintaxe:

```
RangeMode(first_expr {, Expression})
```

Tipo de dados de retorno: numérico

Argumentos:

Os argumentos dessa função podem conter funções interregistro, que por si só retornam um intervalo de valores.

Argumentos

Argumento	Descrição
first_expr	A expressão ou campo que contém os dados a serem medidos.
Expression	Expressões ou campos opcionais que contêm o intervalo de dados a ser medido.

Limitações:

Se houver mais de um valor com a maior frequência, será retornado NULL.

Exemplos e resultados:

Exemplos de funções

Exemplos	Resultados
`RangeMode (1,2,9,2,4)`	Retorna 2
`RangeMode ('a',4,'a',4)`	Retorna NULL
`RangeMode (null( ))`	Retorna NULL

Exemplo:

Adicione o script de exemplo ao seu aplicativo e execute-o. Para ver o resultado, adicione os campos listados na coluna de resultados a uma pasta no seu aplicativo.

```
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
];
```

A tabela resultante mostra os valores retornados de**MyRangeMode** para cada registro na tabela.

Tabela de resultados

RangeID	**MyRangMode**
1	-
2	-
3	8
4	-
5	5
6	-

Exemplo com a expressão:

```
RangeMode (Above(MyField,0,3))
```

Retorna o valor mais comumente ocorrido nos três resultados de **MyField** avaliado na linha atual e duas linhas acima da atual. Ao especificar o terceiro argumento como 3, a função **Above()** retorna três valores, em que há linhas suficientes acima, que são tomadas como entrada na função **RangeMode()**.

Dados usados no exemplo:

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
] ;
```

*Desative a classificação de **MyField** para garantir que o exemplo funcione como esperado.*

Dados de exemplo

MyField	**RangeMode(Above(MyField,0,3))**
10	Retorna 10 porque não existem linhas acima, portanto o valor único é o que ocorre mais comumente.
2	-

MyField	RangeMode(Above(MyField,0,3))
8	-
18	-
5	-
9	-

Consulte também:

Mode - função de gráfico (page 344)

RangeNPV

RangeNPV() retorna o valor líquido atual de um investimento com base em uma taxa de desconto e em uma série de pagamentos periódicos futuros (valores negativos) e receitas (valores positivos). O resultado apresenta um formato numérico padrão de **money**.

Para fluxos de caixa que não são necessariamente periódicos, consulte *RangeXNPV (page 1375)*.

Sintaxe:

```
RangeNPV(discount_rate, value[,value][, Expression])
```

Tipo de dados de retorno: numérico

Argumentos

Argumento	Descrição
discount_rate	A taxa de juros por período.
value	Um pagamento ou a receita que ocorre no final de cada período. Cada valor é um valor único ou um intervalo de valores retornado por uma função inter-registro com um terceiro parâmetro opcional.
Expression	Expressões ou campos opcionais que contêm o intervalo de dados a ser medido.

Limitações:

Os valores de texto, os valores NULL e os valores ausentes são ignorados.

Exemplos	Resultados
`RangeNPV(0.1,-10000,3000,4200,6800)`	Retorna 1188,44

Exemplos	Resultados

Adicione o script de exemplo ao seu aplicativo e execute-o. Para ver o resultado, adicione os campos listados na coluna de resultados a uma pasta no seu aplicativo.
`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 '|');`
	A tabela resultante mostra os valores retornados de RangeNPV para cada um dos registros na tabela.
	RangeID RangeNPV
	1 $-49.13
	2 $777.78
	3 $98.77
	4 $25.51
	5 $250.83
	6 $20.40

Consulte também:

Funções interregistro (page 1278)

RangeNullCount

RangeNullCount() encontra o número de valores NULL na expressão ou campo.

Sintaxe:

```
RangeNullCount(first_expr [, Expression])
```

Tipo de dados de retorno: inteiro

Argumentos:

Os argumentos dessa função podem conter funções interregistro, que por si só retornam um intervalo de valores.

Argumentos

Argumento	Descrição
first_expr	A expressão ou campo que contém os dados a serem medidos.
Expression	Expressões ou campos opcionais que contêm o intervalo de dados a ser medido.

Exemplos e resultados:

Exemplos de funções

Exemplos	Resultados
`RangeNullCount (1,2,4)`	Retorna 0

Exemplos	Resultados
`RangeNullCount (5,'abc')`	Retorna 0
`RangeNullCount (null( ), null( ))`	Retorna 2

Exemplo com a expressão:

```
RangeNullCount (Above(Sum(MyField),0,3))
```

Retorna o número de valores NULL nos três resultados da função **Sum(MyField)** avaliada na linha atual e duas linhas acima da atual.

*Copiar **MyField** no exemplo abaixo não resultará no valor NULL.*

Dados de exemplo

MyField	RangeNullCount(Above(Sum(MyField),0,3))
10	Retorna 2 porque não existem linhas acima desta linha, portanto 2 dos 3 valores estão ausentes (=NULL).
'abc'	Retorna 1 porque existe apenas uma linha acima da linha atual, portanto um dos três valores está ausente (=NULL).
8	Retorna 0 porque nenhuma das três linhas é um valor NULL.

Dados usados nos exemplos:

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
8
] ;
```

Consulte também:

NullCount - função de gráfico (page 360)

RangeNumericCount

RangeNumericCount() encontra o número de valores numéricos em uma expressão ou campo.

Sintaxe:

```
RangeNumericCount(first_expr[, Expression])
```

Tipo de dados de retorno: inteiro

Argumentos:

Os argumentos dessa função podem conter funções interregistro, que por si só retornam um intervalo de valores.

Argumentos

Argumento	Descrição
first_expr	A expressão ou campo que contém os dados a serem medidos.
Expression	Expressões ou campos opcionais que contêm o intervalo de dados a ser medido.

Exemplos e resultados:

Exemplos de funções

Exemplos	Resultados
`RangeNumericCount (1,2,4)`	Retorna 3
`RangeNumericCount (5,'abc')`	Retorna 1
`RangeNumericCount (null( ))`	Retorna 0

Exemplo com a expressão:

```
RangeNumericCount (Above(MaxString(MyField),0,3))
```

Retorna o número de valores numéricos nos três resultados da função **MaxString(MyField)** avaliada na linha atual e duas linhas acima da atual.

*Desative a classificação de **MyField** para garantir que o exemplo funcione como esperado.*

Dados de exemplo

MyField	RangeNumericCount(Above(MaxString(MyField),0,3))
10	1
abc	1
8	2
def	1
xyz	1
9	1

Dados usados nos exemplos:

```
RangeTab:
```

```
LOAD * INLINE [
MyField
10
'abc'
8
def
xyz
9
] ;
```

Consulte também:

- *NumericCount - função de gráfico (page 363)*

RangeOnly

RangeOnly() é uma função dupla que retorna um valor se a expressão for avaliada como um valor único. Se não for esse o caso, **NULL** é retornado.

Sintaxe:

```
RangeOnly(first_expr[, Expression])
```

Tipo de dados de retorno: valor duplo

Argumentos:

Os argumentos dessa função podem conter funções interregistro, que por si só retornam um intervalo de valores.

Argumento	Descrição
first_expr	A expressão ou campo que contém os dados a serem medidos.
Expression	Expressões ou campos opcionais que contêm o intervalo de dados a ser medido.

Exemplos e resultados:

Exemplos	Resultados
`RangeOnly (1,2,4)`	Retorna NULL
`RangeOnly (5,'abc')`	Retorna NULL
`RangeOnly (null( ), 'abc')`	Retorna 'abc'
`RangeOnly(10,10,10)`	Retorna 10

Consulte também:

- *Only - função de gráfico (page 347)*

RangeSkew

RangeSkew() é uma função de gráfico numérica que retorna o valor correspondente à assimetria de um intervalo de números.

Sintaxe:

```
RangeSkew(first_expr[, Expression])
```

Tipo de dados de retorno: numérico

Argumentos:

Os argumentos dessa função podem conter funções interregistro, que por si só retornam um intervalo de valores.

Argumentos

Argumento	Descrição
first_expr	A expressão ou campo que contém os dados a serem medidos.
Expression	Expressões ou campos opcionais que contêm o intervalo de dados a ser medido.

Limitações:

Se não for encontrado nenhum valor numérico, será retornado NULL.

Exemplos e resultados:

Exemplos de funções

Exemplos	Resultados
`rangeskew (1,2,4)`	Retorna 0,93521952958283
`rangeskew (above (SalesValue,0,3))`	Retorna a assimetria variável do intervalo dos três valores retornados da função above() calculada na linha atual e duas linhas acima da atual.

Dados usados no exemplo:

Dados de exemplo

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
] ;
```

Consulte também:

- *Skew - função de gráfico (page 451)*

RangeStdev

RangeStdev() localiza o desvio padrão de um intervalo de números.

Sintaxe:

```
RangeStdev(first_expr[, Expression])
```

Tipo de dados de retorno: numérico

Argumentos:

Os argumentos dessa função podem conter funções interregistro, que por si só retornam um intervalo de valores.

Argumentos

Argumento	Descrição
first_expr	A expressão ou campo que contém os dados a serem medidos.
Expression	Expressões ou campos opcionais que contêm o intervalo de dados a ser medido.

Limitações:

Se não for encontrado nenhum valor numérico, será retornado NULL.

Exemplos e resultados:

Exemplos de funções

Exemplos	Resultados
`RangeStdev (1,2,4)`	Retorna 1,5275252316519
`RangeStdev (null( ))`	Retorna NULL
`RangeStdev (above (SalesValue),0,3))`	Retorna um padrão variável do intervalo dos três valores retornados da função above() calculada na linha atual e duas linhas acima da atual.

Dados usados no exemplo:

Dados de exemplo

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
] ;
```

Consulte também:

- *Stdev - função de gráfico (page 454)*

RangeSum

O **RangeSum()** retorna a soma de um intervalo de valores. Todos os valores não numéricos são tratados como 0.

Sintaxe:

```
RangeSum(first_expr[, Expression])
```

Tipo de dados de retorno: numérico

Argumentos:

Os argumentos dessa função podem conter funções interregistro, que por si só retornam um intervalo de valores.

Argumentos

Argumento	Descrição
first_expr	A expressão ou campo que contém os dados a serem medidos.
Expression	Expressões ou campos opcionais que contêm o intervalo de dados a ser medido.

Limitações:

A função **RangeSum** trata todos os valores não numéricos como 0.

Exemplos e resultados:

Exemplos

Exemplos	Resultados
`RangeSum (1,2,4)`	Retorna 7
`RangeSum (5,'abc')`	Retorna 5
`RangeSum (null( ))`	Retorna 0

Exemplo:

Adicione o script de exemplo ao seu aplicativo e execute-o. Para ver o resultado, adicione os campos listados na coluna de resultados a uma pasta no seu aplicativo.

```
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
];
```

A tabela resultante mostra os valores retornados de MyRangeSum para cada um dos registros na tabela.

Tabela resultante

RangeID	MyRangeSum
1	21
2	12
3	18
4	38
5	19
6	15

Exemplo com a expressão:

```
RangeSum (Above(MyField,0,3))
```

Retorna a soma de três valores de **MyField)**: na linha atual e duas linhas acima da atual. Ao especificar o terceiro argumento como 3, a função **Above()** retorna três valores, em que há linhas suficientes acima, que são tomadas como entrada na função **RangeSum()**.

Dados usados nos exemplos:

*Desative a classificação de **MyField** para garantir que o exemplo funcione como esperado.*

Dados de exemplo

MyField	RangeSum(Above(MyField,0,3))
10	10
2	12
8	20
18	28
5	31
9	32

Dados usados nos exemplos:

```
RangeTab:
```

```
LOAD * INLINE [
MyField
10
2
8
18
5
9
] ;
```

Consulte também:

- *Sum - função de gráfico (page 349)*
- *Above - função de gráfico (page 1282)*

RangeTextCount

RangeTextCount() retorna o número de valores de texto em uma expressão ou campo.

Sintaxe:

```
RangeTextCount(first_expr[, Expression])
```

Tipo de dados de retorno: inteiro

Argumentos:

Os argumentos dessa função podem conter funções interregistro, que por si só retornam um intervalo de valores.

Argumento

Argumento	Descrição
first_expr	A expressão ou campo que contém os dados a serem medidos.
Expression	Expressões ou campos opcionais que contêm o intervalo de dados a ser medido.

Exemplos e resultados:

Exemplos de funções

Exemplos	Resultados
`RangeTextCount (1,2,4)`	Retorna 0
`RangeTextCount (5,'abc')`	Retorna 1
`RangeTextCount (null( ))`	Retorna 0

Exemplo com a expressão:

```
RangeTextCount (Above(MaxString(MyField),0,3))
```

Retorna o número de valores de texto dentro dos três resultados da função **MaxString(MyField)** avaliada na linha atual e duas linhas acima da atual.

Dados usados nos exemplos:

*Desative a classificação de **MyField** para garantir que o exemplo funcione como esperado.*

Dados de exemplo

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

Dados usados nos exemplos:

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
8
null()
'xyz'
9
] ;
```

Consulte também:

TextCount - função de gráfico (page 367)

RangeXIRR

RangeXIRR() retorna a taxa interna de retorno (anual) para uma programação de fluxos de caixa que não é necessariamente periódica. Para calcular a taxa de retorno interno de uma série de fluxos de caixa periódicos, use a função **RangeIRR**.

Funcionalidade XIRR do Qlik (funções **XIRR()** e **RangeXIRR()**) usa a seguinte equação, resolvendo para o valor `Rate`, para determinar o valor XIRR correto:

```
XNPV(Rate, pmt, date) = 0
```

A equação é resolvida usando uma versão simplificada do método de Newton.

Sintaxe:

```
RangeXIRR(value, date{, value, date})
```

Tipo de dados de retorno: numérico

Argumentos

Argumento	Descrição
value	Um fluxo de caixa ou uma série de fluxos de caixa correspondente a uma programação de pagamentos nas datas. A série de valores deve conter, pelo menos, um valor positivo e um valor negativo.
date	Uma data de pagamento ou uma programação de datas de pagamento correspondente aos pagamentos do fluxo de caixa.

Ao trabalhar com essa função, as seguintes limitações são aplicáveis:

- Os valores de texto, os valores NULL e os valores ausentes são ignorados.
- Todos os pagamentos têm descontos baseados em um ano de 365 dias.
- Essa função requer pelo menos um pagamento negativo válido e pelo menos um pagamento positivo válido (com datas válidas correspondentes). Se esses pagamentos não forem fornecidos, um valor NULL será retornado.

Os tópicos a seguir podem ajudar você a trabalhar com essa função:

- *RangeXNPV (page 1375)*: Use essa função para calcular o valor presente líquido para uma programação de fluxos de caixa que não seja necessariamente periódica.
- *XIRR (page 381)*: A função **XIRR()** calcula a taxa interna de retorno agregada (anualmente) para uma programação de fluxos de caixa (que não é necessariamente periódica).

Em diferentes versões do Qlik Sense Client-Managed, há variações no algoritmo subjacente usado por essa função. Para obter mais informações sobre atualizações recentes do algoritmo, consulte o artigo de suporte Correção e atualização da função XIRR.

Exemplos e resultados:

Exemplos e resultados

Exemplos	Resultados
`RangeXIRR(-2500,'2008-01-01',2750,'2008-09-01')`	Retorna 0,1532

Consulte também:

- *RangeIRR (page 1349)*
- *RangeXNPV (page 1375)*
- *XIRR (page 381)*

Correção e atualização da função XIRR

RangeXNPV

RangeXNPV() retorna o valor presente líquido para um cronograma de fluxos de caixa (não necessariamente periódicos) representados por números emparelhados nas expressões especificadas por **pmt** e **date**. Todos os pagamentos têm descontos baseados em um ano de 365 dias.

Sintaxe:

```
RangeXNPV(discount_rate, value, date{, value, date})
```

Tipo de dados de retorno: numérico

Argumentos

Argumento	Descrição
discount_rate	**discount_rate** é a taxa anual com base na qual os pagamentos devem ser descontados.
value	Um fluxo de caixa ou uma série de fluxos de caixa correspondente a uma programação de pagamentos nas datas. Cada valor é um valor único ou um intervalo de valores retornado por uma função inter-registro com um terceiro parâmetro opcional. A série de valores deve conter, pelo menos, um valor positivo e um valor negativo.
date	Uma data de pagamento ou uma programação de datas de pagamento correspondente aos pagamentos do fluxo de caixa.

Ao trabalhar com essa função, as seguintes limitações são aplicáveis:

- Os valores de texto, os valores NULL e os valores ausentes são ignorados.
- Todos os pagamentos têm descontos baseados em um ano de 365 dias.

Exemplo - script

Script de carregamento e resultados

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Dados financeiros contidos em uma tabela chamada `RangeTab3`.
- O uso da função **RangeXNPV()** para calcular o valor presente líquido.

Script de carregamento

```
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 '|');
```

Resultados

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione esses campos como dimensões:

- `RangeID`
- `RangeXNPV`

Tabela de resultados

RangeID	RangeXNPV
1	-$9.09
2	-$0.00
3	$13.64

Exemplo - expressão de gráfico

Script de carregamento e expressão de gráfico

Visão geral

Abra o editor da carga de dados e adicione o script de carregamento abaixo em uma nova guia.

O script de carregamento contém:

- Dados financeiros contidos em uma tabela chamada `RangeTab3`.
- O uso da função **RangeXNPV()** para calcular o valor presente líquido.

Script de carregamento

```
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 '|');
```

Resultados

Faça o seguinte:

Carregue os dados e abra uma pasta. Crie uma nova tabela e adicione o seguinte cálculo como medida:

```
=RangeXNPV(0.1, -2500,'2008-01-01',2750,'2008-09-01')
```

Tabela de resultados

=XIRR(Payments, Date)
$80.25

Consulte também:

XNPV (page 388)

5.22 Funções relacionais

Esse é um grupo de funções que calculam propriedades de valores dimensionais individuais em um gráfico usando números já agregados.

As funções são relacionais no sentido de que a saída da função depende não apenas do valor do ponto de dados propriamente dito, mas também da relação do valor com outros pontos de dados. Por exemplo, uma classificação não pode ser calculada sem uma comparação com outros valores dimensionais.

Essas funções só podem ser usadas em expressões de gráficos. Elas não podem ser usadas no script de carregamento.

É necessária uma dimensão no gráfico, pois isso define os outros pontos de dados necessários para a comparação. Consequentemente, uma função relacional não é significativa em um gráfico sem dimensão (por exemplo, um objeto KPI).

Funções de classificação

A omissão de valores zero é automaticamente desabilitada quando essas funções são usadas. Os valores NULL são desconsiderados.

Rank

Rank() avalia as linhas do gráfico na expressão e, para cada linha, exibe a posição relativa do valor da dimensão avaliada na expressão. Ao avaliar a expressão, a função compara o resultado com o resultado de outras linhas que contêm o segmento de coluna atual e retorna a posição da linha atual dentro do segmento.

```
Rank - função de gráfico([TOTAL [<fld {, fld}>]] expr[, mode[, fmt]])
```

HRank

HRank() avalia a expressão, e compara o resultado com o resultado de outras colunas que contêm o segmento de linha atual de uma tabela dinâmica. A função então retorna a classificação da coluna atual dentro do segmento.

```
HRank - função de gráfico([TOTAL] expr[, mode[, fmt]])
```

Funções de agrupamento

KMeans2D

O grupo de propriedades **Licença do site** contém propriedades relacionadas à licença do sistema Qlik Sense. Todos os campos são obrigatórios e não devem estar vazios.

Propriedades de licença de site

Nome da propriedade	Descrição
Nome do proprietário	O nome de usuário do proprietário do produto Qlik Sense.
Organização do proprietário	O nome da organização da qual o proprietário do produto Qlik Sense é membro.
Número de série	O número de série atribuído ao software Qlik Sense.
Número de controle	O número de controle atribuído ao software Qlik Sense.
Acesso ao LEF	O License Enabler File (LEF) atribuído ao software Qlik Sense.

O **KMeans2D()** avalia as linhas do gráfico por meio da aplicação do agrupamento de k-means e, para cada linha do gráfico, exibe o ID do agrupamento ao qual esse ponto de dados foi atribuído. As colunas que são usadas pelo algoritmo de agrupamento são determinadas pelos parâmetros coordinate_1 e coordinate_2, respectivamente. Ambas são agregações. O número de agrupamentos criados é determinado pelo parâmetro num_clusters. Os dados podem ser normalizados opcionalmente pelo parâmetro norm.

```
KMeans2D - função de gráfico(num_clusters, coordinate_1, coordinate_2 [,
norm])
```

KMeansND

O **KMeansND()** avalia as linhas do gráfico por meio da aplicação do agrupamento de k-means e, para cada linha do gráfico, exibe o ID do agrupamento ao qual esse ponto de dados foi atribuído. As colunas que são usadas pelo algoritmo de agrupamento são determinadas pelos parâmetros coordinate_1 e coordinate_2, etc., até n colunas. Todas são agregações. O número de agrupamentos criados é determinado pelo parâmetro num_clusters.

```
KMeansND - função de gráfico(num_clusters, num_iter, coordinate_1,
coordinate_2 [,coordinate_3 [, ...]])
```

KMeansCentroid2D

O **KMeansCentroid2D()** avalia as linhas do gráfico por meio da aplicação do agrupamento de k-means e, para cada linha do gráfico, exibe a coordenada desejada do agrupamento ao qual esse ponto de dados foi atribuído. As colunas que são usadas pelo algoritmo de agrupamento são determinadas pelos parâmetros coordinate_1 e coordinate_2, respectivamente. Ambas são agregações. O número de agrupamentos criados é determinado pelo parâmetro num_clusters. Os dados podem ser normalizados opcionalmente pelo parâmetro norm.

```
KMeansCentroid2D - função de gráfico(num_clusters, coordinate_no, coordinate_
1, coordinate_2 [, norm])
```

KMeansCentroidND

O **KMeansCentroidND()** avalia as linhas do gráfico por meio da aplicação do agrupamento de k-means e, para cada linha do gráfico, exibe a coordenada desejada do agrupamento ao qual esse ponto de dados foi atribuído. As colunas que são usadas pelo algoritmo de agrupamento são determinadas pelos parâmetros coordinate_1, coordinate_2, etc., até n colunas. Todas são agregações. O número de agrupamentos criados é determinado pelo parâmetro num_clusters.

```
KMeansCentroidND - função de gráfico(num_clusters, num_iter, coordinate_no,
coordinate_1, coordinate_2 [,coordinate_3 [, ...]])
```

Funções de decomposição de séries temporais

STL_Trend

STL_Trend é uma função de decomposição de séries temporais. Junto com **STL_Seasonal** e **STL_Residual**, essa função é usada para decompor uma série temporal em componentes sazonais, de tendências e residuais. No contexto do algoritmo STL, a decomposição de séries temporais é usada para identificar tanto um padrão sazonal recorrente quanto uma tendência geral, considerando uma métrica de entrada e outros parâmetros. A função **STL_Trend** identificará uma tendência geral, independente dos padrões ou ciclos sazonais, a partir de dados de séries temporais.

```
STL_Trend - função de gráfico(target_measure, period_int [,seasonal_smoother
[,trend_smoother]])
```

STL_Seasonal

STL_Seasonal é uma função de decomposição de séries temporais. Junto com **STL_Trend** e **STL_Residual**, essa função é usada para decompor uma série temporal em componentes sazonais, de tendências e residuais. No contexto do algoritmo STL, a decomposição de séries temporais é usada para identificar tanto um padrão sazonal recorrente quanto uma tendência geral, considerando uma métrica de entrada e outros parâmetros. A função **STL_Seasonal** pode identificar um padrão sazonal em uma série temporal, separando-o da tendência geral exibida pelos dados.

```
STL_Seasonal - função de gráfico(target_measure, period_int [,seasonal_
smoother [,trend_smoother]])
```

STL_Residual

STL_Residual é uma função de decomposição de séries temporais. Juntamente com **STL_Seasonal** e **STL_Trend**, essa função é usada para decompor uma série temporal em componentes sazonais, de tendência e residuais. No contexto do algoritmo STL, a decomposição de séries temporais é usada para identificar tanto um padrão sazonal recorrente quanto uma tendência geral, considerando uma métrica de entrada e outros parâmetros. Ao realizar essa operação, parte da variação na métrica de entrada não se encaixará no componente sazonal nem no componente de tendências e será definida como o componente residual. A função de gráfico **STL_Residual** captura essa parte do cálculo.

```
STL_Residual - função de gráfico(target_measure, period_int [,seasonal_
smoother [,trend_smoother]])
```

Rank - função de gráfico

Rank() avalia as linhas do gráfico na expressão e, para cada linha, exibe a posição relativa do valor da dimensão avaliada na expressão. Ao avaliar a expressão, a função compara o resultado com o resultado de outras linhas que contêm o segmento de coluna atual e retorna a posição da linha atual dentro do segmento.

Segmentos de coluna

	Region	Country	Population	Rank(Population)
Column segment #1	Americas	Mexico	128,932,753	2
	Americas	Canada	37,742,154	3
	Americas	United States of America	331,002,651	1
Column segment #2	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

Para gráficos que não sejam tabelas, o segmento de coluna atual é definido conforme ele é exibido no equivalente de tabela estática do gráfico.

Sintaxe:

```
Rank([TOTAL] expr[, mode[, fmt]])
```

Tipo de dados de retorno: dual

Argumentos:

Argumentos

Argumento	Descrição
expr	A expressão ou campo que contém os dados a serem medidos.
mode	Especifica a representação numérica do resultado da função.
fmt	Especifica a representação textual do resultado da função.
TOTAL	Se o gráfico for unidimensional ou se a expressão for precedida pelo qualificador **TOTAL**, a função será avaliada ao longo da coluna inteira. Se a tabela ou o equivalente de tabela tiver várias dimensões verticais, o segmento de coluna atual incluirá somente linhas com os mesmos valores que a linha atual em todas as colunas de dimensão, exceto na coluna que mostrar a última dimensão na ordem de classificação entre os campos.

A posição é retornada como um valor dual, que, quando cada linha tiver uma posição única, será um número inteiro entre 1 e o número de linhas do segmento da coluna atual.

Caso várias linhas compartilhem a mesma posição, a representação textual e numérica poderá ser controlada com os parâmetros **mode** e **fmt**.

mode

O segundo argumento, **mode**, pode ter os seguintes valores:

Exemplos de **mode**

Valor	Descrição
0 (padrão)	Se todas as posições dentro do grupo de compartilhamento estiverem no lado inferior do valor intermediário da posição, todas as linhas receberão a posição mais baixa dentro do grupo. Se todas as posições de compartilhamento dentro do grupo estiverem no lado superior do valor intermediário da posição, todas as linhas receberão a posição mais alta dentro do grupo. Se as posições dentro do grupo se estenderem além do valor intermediário de toda a posição, todas as linhas receberão o valor correspondente à média da posição superior e inferior de todo o segmento de coluna.
1	Posição mais baixa em todas as linhas.
2	Posição média em todas as linhas.
3	Posição mais alta em todas as linhas.
4	Posição mais baixa na primeira linha, depois, aumentada em incrementos de um para cada linha.

fmt

O terceiro argumento, **fmt**, pode ter os seguintes valores:

Exemplos de **fmt**

Valor	Descrição
0 (padrão)	Valor baixo **-** valor alto em todas as linhas (por exemplo 3-4).
1	Valor baixo em todas as linhas.
2	Valor baixo na primeira linha, em branco nas linhas seguintes.

A ordem das linhas para **mode** 4 e **fmt** 2 é determinada pela ordem de classificação das dimensões do gráfico.

Exemplos e resultados:

Crie duas visualizações a partir das dimensões Product e Sales e outra a partir de Product e UnitSales. Adicione medidas, conforme mostrado na tabela a seguir.

Exemplos de Rank

Exemplos	Resultados
Exemplo 1. Crie uma tabela com as dimensões `Customer` e `Sales` e com a medida `Rank(Sales)`	O resultado dependerá da ordem de classificação das dimensões. Se a tabela for classificada em Customer, a tabela listará todos os valores de Sales para Astrida, Betacab, e assim por diante. Os resultados de Rank(Sales) mostrarão 10 para o valor de 12 de Sales, 9 para o valor de 13 de Sales e assim por diante, com o valor da posição de 1 retornado para o valor 78 de Sales. O próximo segmento da coluna começa com Betacab, para o qual o primeiro valor de Sales no segmento é 12. O valor da posição de Rank(Sales) para este é dado como 11. Se a tabela for classificada em Sales, os segmentos da coluna consistirão nos valores de Sales e do Customer correspondente. Como existem dois valores de 12 de Sales (para Astrida e Betacab), o valor de Rank(Sales) para esse segmento de coluna é 1-2, para cada valor de Customer. Isso ocorre pois existem dois valores de Customer para o valor de 12 de Sales. Se existissem 4 valores, o resultado seria 1-4 para todas as linhas. Isso mostra qual seria o resultado para o valor padrão (0) do argumento fmt.
Exemplo 2. Substitua a dimensão Customer por Product e adicione a medida `Rank(Sales,1,2)`	Isto retorna 1 na primeira linha em cada segmento de coluna e deixa todas as outras linhas em branco, pois os argumentos **mode** e **fmt** são definidos para 1 e 2, respectivamente.

Resultados para o exemplo 1, com a tabela classificada por Customer:

Tabela de resultados

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

Resultados para o exemplo 1, com a tabela classificada por Sales:

Tabela de resultados

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

Dados usados nos exemplos:

```
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 '|');
```

Consulte também:

- *Sum - função de gráfico (page 349)*

HRank - função de gráfico

HRank() avalia a expressão, e compara o resultado com o resultado de outras colunas que contêm o segmento de linha atual de uma tabela dinâmica. A função então retorna a classificação da coluna atual dentro do segmento.

Sintaxe:

```
HRank([ TOTAL ] expr [ , mode [, fmt ] ])
```

Tipo de dados de retorno: dual

> *Esta função só funciona em tabelas dinâmicas. Em todos os outros tipos de gráfico, ela retorna NULL.*

Argumentos:

Argumentos

Argumento	Descrição
expr	A expressão ou campo que contém os dados a serem medidos.
mode	Especifica a representação numérica do resultado da função.
fmt	Especifica a representação textual do resultado da função.
TOTAL	Se o gráfico for unidimensional ou se a expressão for precedida pelo qualificador **TOTAL**, a função será avaliada ao longo da coluna inteira. Se a tabela ou o equivalente de tabela tiver várias dimensões verticais, o segmento de coluna atual incluirá somente linhas com os mesmos valores que a linha atual em todas as colunas de dimensão, exceto na coluna que mostrar a última dimensão na ordem de classificação entre os campos.

Se a tabela dinâmica for unidimensional ou se a expressão for precedida pelo qualificador **total**, o segmento de linha atual será sempre igual à linha inteira. Se a tabela dinâmica tiver várias dimensões horizontais, o segmento de linha atual incluirá somente colunas com os mesmos valores que a coluna atual em todas as linhas de dimensão, exceto na linha que mostrar a última dimensão horizontal na ordem de classificação entre os campos.

A posição é retornada como um valor dual, que, quando cada coluna tiver uma posição única, será um número inteiro entre 1 e o número de colunas do segmento de linha atual.

Caso várias colunas compartilhem a mesma posição, a representação textual e numérica poderá ser controlada com os parâmetros **mode** e **format**.

O segundo argumento, **mode**, especifica a representação numérica do resultado da função:

Exemplos de **mode**

Valor	Descrição
0 (padrão)	Se todas as posições de compartilhamento dentro do grupo de compartilhamento estiverem no lado inferior do valor intermediário da posição, todas as colunas receberão a posição mais baixa dentro do grupo. Se todas as posições de compartilhamento dentro do grupo estiverem no lado superior do valor intermediário da posição, todas as colunas receberão a posição mais alta dentro do grupo. Se as posições dentro do grupo se estenderem além do valor intermediário de toda a posição, todas as linhas receberão o valor correspondente à média da posição superior e inferior de todo o segmento de coluna.
1	Posição mais baixa em todas as colunas do grupo.
2	Posição média em todas as colunas do grupo.
3	Posição mais alta em todas as colunas do grupo.
4	Posição mais baixa na primeira coluna, depois, aumentada em incrementos de um para cada coluna do grupo.

O terceiro argumento, **format**, especifica a representação textual do resultado da função:

Exemplos de **format**

Valor	Descrição
0 (padrão)	Valor baixo**&' - '&**valor alto em todas as colunas no grupo (por exemplo, 3 - 4).
1	Valor baixo em todas as colunas do grupo.
2	Valor baixo na primeira coluna, em branco nas colunas seguintes do grupo.

A ordem das colunas para **mode** 4 e **format** 2 é determinada pela ordem de classificação das dimensões do gráfico.

Exemplos:

```
HRank( sum( Sales ))
```

```
HRank( sum( Sales ), 2 )
```

```
HRank( sum( Sales ), 0, 1 )
```

Otimização com o k-means: Um exemplo do mundo real

O exemplo a seguir ilustra um caso de uso do mundo real em que o agrupamento KMeans e as funções de Centroide são aplicados a um conjunto de dados. A função KMeans separa os pontos de dados em agrupamentos que compartilham semelhanças. Os agrupamentos tornam-se mais compactos e diferenciados à medida que o algoritmo KMeans é aplicado em um número configurável de iterações.

O KMeans é usado em muitos campos em uma ampla variedade de casos de uso. Alguns exemplos de casos de uso de agrupamento incluem segmentação de clientes, detecção de fraudes, previsão de atritos de conta, direcionamento de incentivos ao cliente, identificação de criminosos cibernéticos e otimização de rotas de entrega. O algoritmo de agrupamento KMeans está cada vez mais sendo usado para casos em que as empresas tentam inferir padrões e otimizar ofertas de serviços.

Qlik Sense Funções KMeans e de Centroide

O Qlik Sense fornece duas funções KMeans que agrupam pontos de dados em agrupamentos com base na semelhança. Consulte *KMeans2D - função de gráfico (page 1395)* e *KMeansND - função de gráfico (page 1410)*. A função **KMeans2D** aceita duas dimensões e funciona bem para visualizar os resultados por meio de um **gráfico de dispersão**. A função **KMeansND** aceita mais de duas dimensões. Como é fácil conceituar um resultado 2D em gráficos padrão, a seguinte demonstração aplicará KMeans em um **gráfico de dispersão** usando duas dimensões. O agrupamento KMeans pode ser visualizado por meio da aplicação de cores por expressão ou por dimensão, conforme descrito neste exemplo.

As funções de centroide do Qlik Sense determinam a posição média aritmética de todos os pontos de dados no agrupamento e identificam um ponto central, ou centroide, para esse agrupamento. Para cada linha (ou registro) do gráfico, a função de centroide exibe a coordenada do agrupamento ao qual esse ponto de dados foi atribuído. Consulte *KMeansCentroid2D - função de gráfico (page 1425)* e *KMeansCentroidND - função de gráfico (page 1427)*.

Visão geral de exemplos e casos de uso

O exemplo a seguir apresenta um cenário simulado do mundo real. Uma empresa têxtil no estado de Nova York, EUA, deve diminuir as despesas minimizando os custos de entrega. Uma maneira de fazer isso é realocar os armazéns que estão mais próximos dos seus distribuidores. A empresa emprega 118 distribuidores em todo o estado de Nova York. A demonstração a seguir simula como um gerente de operações poderia segmentar distribuidores em cinco áreas geográficas agrupadas usando a função KMeans e, em seguida, identificar cinco localizações de armazém ideais e centrais para esses agrupamentos usando a função de centroide. O objetivo é descobrir coordenadas de mapeamento que podem ser usadas para identificar cinco locais de armazém central.

O conjunto de dados

O conjunto de dados é baseado em nomes e endereços gerados aleatoriamente no estado de Nova York com coordenadas reais de latitude e longitude. O conjunto de dados contém as seguintes dez colunas: id, first_ name, last_name, phone, address, city, state, zip, latitude, longitude. O conjunto de dados está disponível abaixo como um arquivo que você pode baixar localmente e, em seguida, fazer upload para o Qlik Sense ou inline no editor de carregamento de dados. O aplicativo que está sendo criado é chamado *Distribuidores KMeans e centroide*, e sua primeira pasta é chamada *Análise de agrupamentos de distribuição*.

Selecione o seguinte link para baixar o arquivo de dados de amostra: DistributorData.csv

Conjunto de dados do Distributor: Carregamento inline para o editor de carregamento de dados no Qlik Sense (page 1392)

Título: DistributorData

Número total de registros: 118

Aplicando a função KMeans2D

Neste exemplo, a configuração de um gráfico de **diagrama de dispersão** é demonstrada usando o conjunto de dados *DistributorData*, a função **KMeans2D** é aplicada, e o gráfico é colorido por dimensão.

Observe que as funções KMeans do Qlik Sense oferecem suporte para agrupamento automático usando um método chamado de diferença de profundidade (DeD). Quando um usuário define 0 para o número de agrupamentos, um número ideal de agrupamentos para esse conjunto de dados é determinado. Para esse exemplo, no entanto, uma variável é criada para o argumento **num_clusters** (consulte *KMeans2D - função de gráfico (page 1395)* para conhecer a sintaxe). Portanto, o número desejado de agrupamentos (k=5) é especificado por uma variável.

1. Um **gráfico de dispersão** é arrastado para a pasta e nomeado *Distribuidores (por dimensão)*.
2. Uma **variável** é criada para especificar o número de agrupamentos. A **variável** se chama *vDistClusters*. Para a variável **Definição**, *5* é inserida.
3. Configuração de **dados** para o gráfico:
    a. Em **Dimensões**, o campo *id* é selecionado para **Bolha**. *ID do agrupamento* é inserido para o **Rótulo**.
    b. Em **Medidas**, *Média([latitude])* é a expressão para **Eixo X**.
    c. Em **Medidas**, *Média([longitude])* é a expressão para **Eixo Y**.
4. Configuração de **Aparência**:
    a. Em **Cores e legenda**, a opção **Personalizado** foi escolhida para **Cores**.
    b. A opção **Por dimensão** foi selecionada para colorir o gráfico.
    c. A seguinte expressão é inserida: *=pick(aggr(KMeans2D(vDistClusters,only(latitude),only (longitude)),id)+1, 'Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster 4', 'Cluster 5')*
    d. A caixa de seleção para **Cores persistentes** está marcada.

Gráfico de dispersão antes que as cores KMeans por dimensão sejam aplicadas

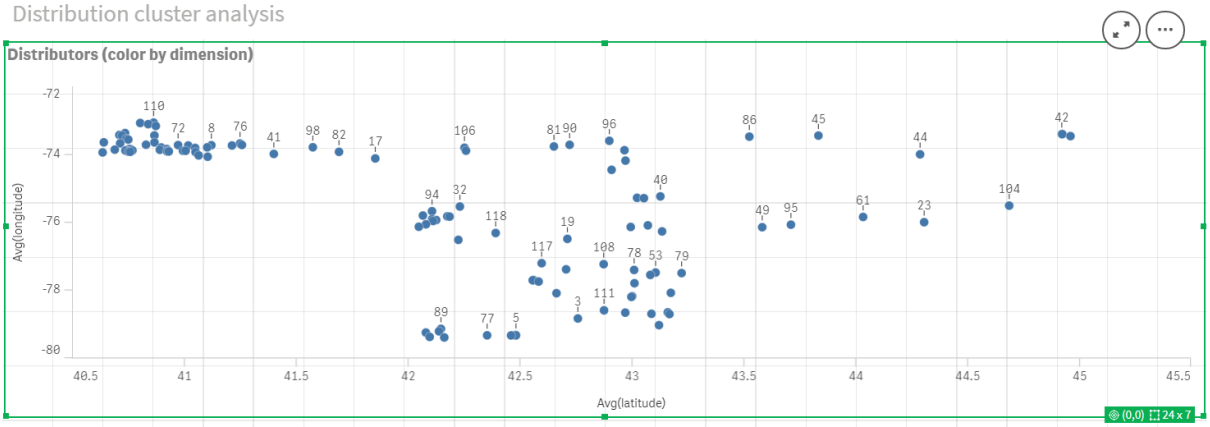
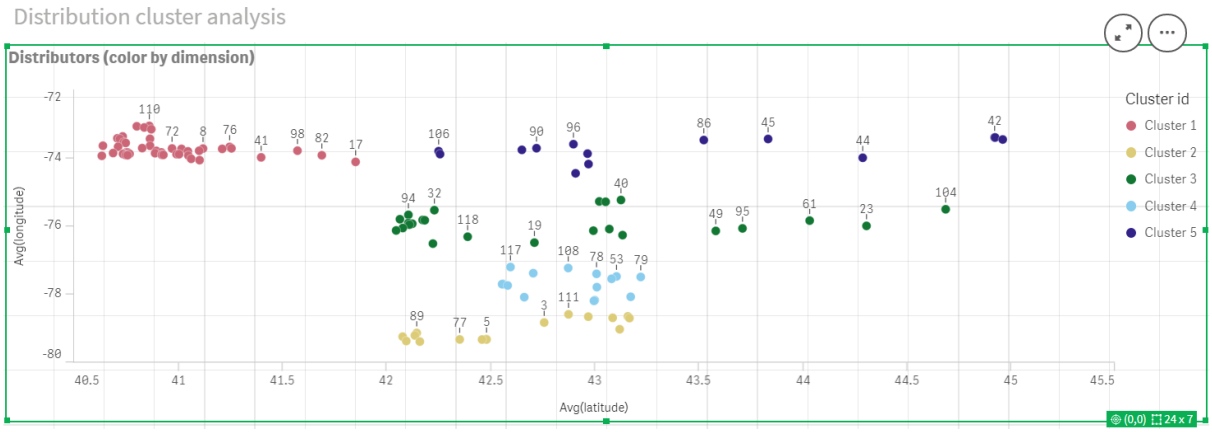


Gráfico de dispersão depois que as cores KMeans por dimensão são aplicadas



Adicionando uma **tabela**: *Distribuidores*

Pode ser útil ter uma tabela à mão para acesso rápido aos dados relevantes. O **gráfico de dispersão** mostra *ids*, embora uma tabela com nomes de distribuidores correspondentes seja adicionada para referência.

1. Uma **tabela** chamada *Distribuidores* é arrastada para a pasta com as seguintes **Colunas** (Dimensões) adicionadas: *id*, *first_name* e *last_name*.

Tabela: Nomes de distribuidor

Distributors

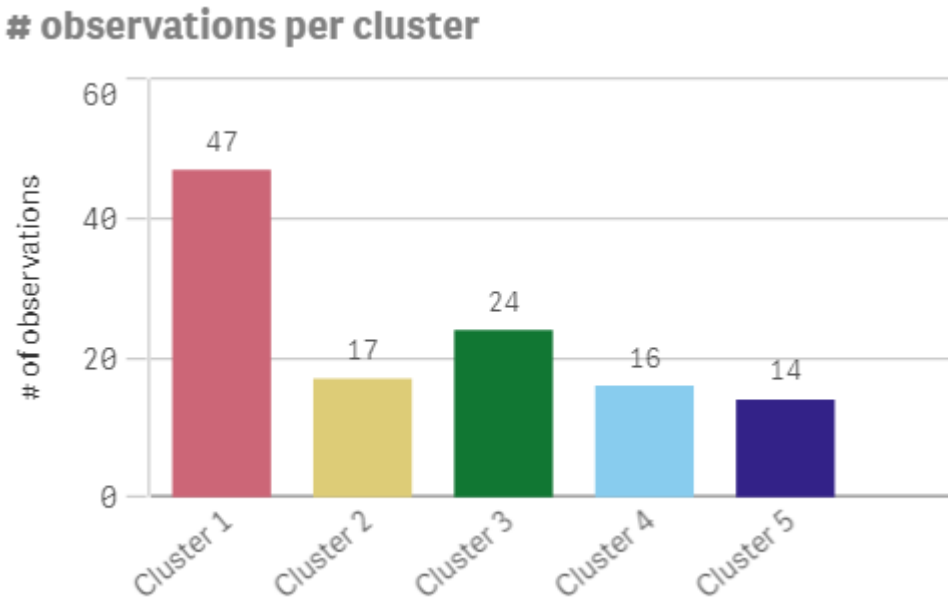
id	first_name	last_name
1	Kaiya	Snow
2	Dean	Roy
3	Eden	Paul
4	Bryanna	Higgins
5	Elisabeth	Lee
6	Skylar	Robinson
7	Cody	Bailey
8	Dario	Sims
9	Deacon	Hood

Adicionando um **gráfico de barras**: *nº de observações por agrupamento*

Para o cenário de distribuição de armazém, é útil saber quantos distribuidores serão atendidos por cada armazém. Portanto, é criado um **gráfico de barras** que mede quantos distribuidores são atribuídos a cada agrupamento.

1. Um **gráfico de barras** é arrastado para a pasta. O gráfico é nomeado: *nº de observações por agrupamento*.
2. Configuração de **dados** para o **gráfico de barras**:
    a. Uma **Dimensão** rotulada *Agrupamentos* é adicionada (o rótulo pode ser adicionado após a aplicação da expressão). A seguinte expressão é inserida: *=pick(aggr(KMeans2D (vDistClusters,only(latitude),only(longitude)),id)+1, 'Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster 4', 'Cluster 5')*
    b. Uma **Medida** rotulada *nº de observações* é adicionada. A seguinte expressão é inserida: *=count (aggr(KMeans2D(vDistClusters,only(latitude),only(longitude)),id))*
3. Configuração de **Aparência**:
    a. Em **Cores e legenda**, a opção **Personalizado** foi escolhida para **Cores**.
    b. A opção **Por dimensão** foi selecionada para colorir o gráfico.
    c. A seguinte expressão é inserida: *=pick(aggr(KMeans2D(vDistClusters,only(latitude),only (longitude)),id)+1, 'Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster 4', 'Cluster 5')*
    d. A caixa de seleção para **Cores persistentes** está marcada.
    e. **Mostrar legenda** é desativado.
    f. Em **Apresentação**, **Rótulos de valor** é alternado para **Auto**.
    g. Em **Eixo X**: **Agrupamentos**, **Apenas rótulos** é selecionado.

Gráfico de barras: nº de observações por agrupamento



Aplicando a função **Centroid2D**

É adicionada uma segunda tabela para a função **Centroid2D** que identificará as coordenadas para locais de armazém em potencial. Esta tabela mostra a localização central (valores de centroide) para os cinco grupos de distribuidores identificados.

1. Uma **Tabela** é arrastada para a pasta e nomeada *Centroides de agrupamento* com as seguintes colunas adicionadas:
    a. Uma **Dimensão** rotulada *Agrupamentos* é adicionada. A seguinte expressão é inserida:=*pick (aggr(KMeans2D(vDistClusters,only(latitude),only(longitude)),id)+1,'Warehouse 1','Warehouse 2','Warehouse 3','Warehouse 4','Warehouse 5'*)
    b. Uma **Medida** rotulada *latitude (D1)* é adicionada. A seguinte expressão é inserida:=*only(aggr (KMeansCentroid2D(vDistClusters,0,only(latitude),only(longitude)),id))*
    Observe que o parâmetro **coordinate_no** corresponde à primeira dimensão(0). Nesse caso, a dimensão *latitude* é traçada em relação ao eixo x. Se estivéssemos trabalhando com a função **CentroidND** e houvesse até seis dimensões, essas entradas de parâmetro poderiam ser qualquer um dos seis valores: 0,1,2,3,4 ou 5.
    c. Uma **Medida** rotulada *longitude (D2)* é adicionada. A seguinte expressão é inserida:=*only(aggr (KMeansCentroid2D(vDistClusters,1,only(latitude),only(longitude)),id))*
    O parâmetro **coordinate_no** nesta expressão corresponde à segunda dimensão(1). A dimensão *longitude* é traçada em relação ao eixo y.

Tabela: Cálculos de centroide de agrupamento

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

Mapeamento de centroide

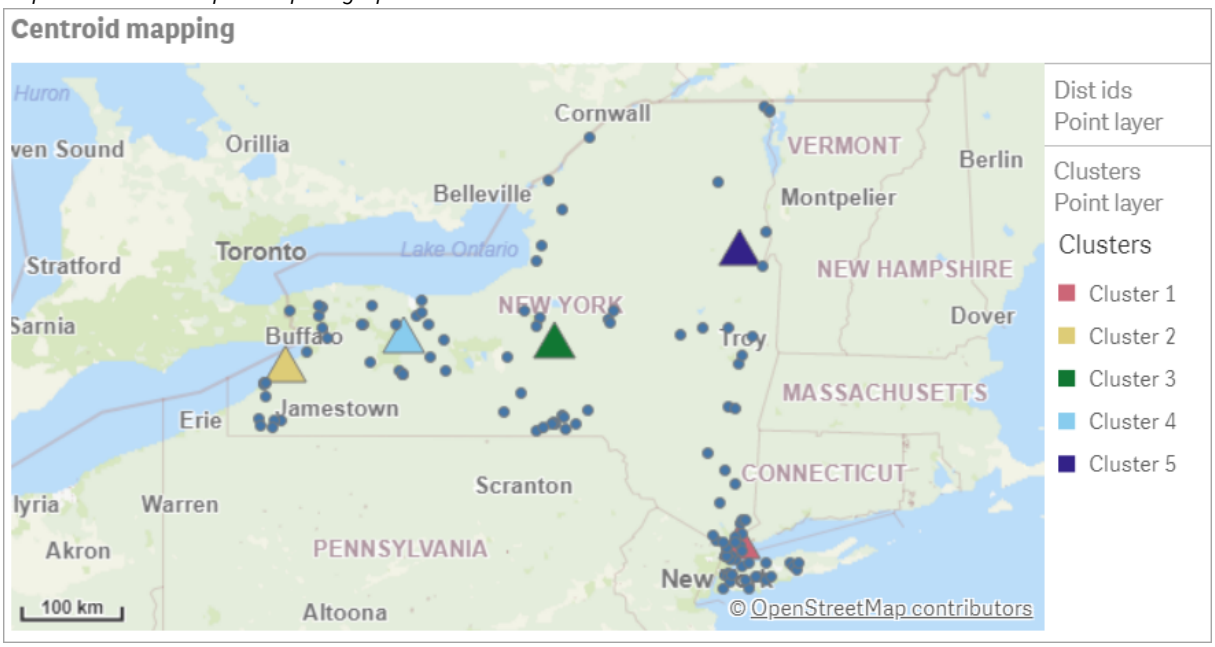
A próxima etapa é mapear os centroides. Cabe ao desenvolvedor do aplicativo decidor se ele prefere colocar a visualização em pastas separadas.

1. Um **mapa** nomeado *Mapeamento de centroide* é arrastado para a pasta.
2. Na seção **Camadas**, **Adicionar camada** é selecionado, depois **Camada de ponto** é selecionado.
    a. O **Campo** *id* é selecionado, e *IDs de distribuição* **Rótulo** é adicionado.
    b. Na seção **Local**, a caixa de seleção para **Campos Latitude e Longitude** é marcada.
    c. Para **Latitude**, o campo *latitude* é selecionado.
    d. Para **Longitude**, o campo *longitude* é selecionado.
    e. Na seção **Tamanho e forma**, **Bolha** é selecionado para **Forma**, e o **Tamanho** é diminuído no controle deslizante de preferência.

f. Na seção **Cores**, **Cor única** é selecionado, e azul é selecionado para a **Cor** e cinza para a cor de **Contorno** (essas escolhas também são uma questão de preferência).

3. Na seção **Camadas**, uma segunda **Camada de ponto** é adicionada selecionando **Adicionar camada** e depois selecionando **Camada de ponto**.
   a. A seguinte expressão é inserida: *=aggr(KMeans2D(vDistClusters,only(latitude),only(longitude)),id)*
   b. O **Rótulo** *Agrupamentos* é adicionado.
   c. Na seção **Local**, a caixa de seleção para **Campos Latitude e Longitude** é marcada.
   d. Para **Latitude**, que, nesse caso, é traçado ao longo do eixo x, a seguinte expressão é adicionada: *=aggr(KMeansCentroid2D(vDistClusters,0,only(latitude),only(longitude)),id)*
   e. Para **Longitude**, que, nesse caso, é traçado ao longo do eixo y, a seguinte expressão é adicionada: *=aggr(KMeansCentroid2D(vDistClusters,1,only(latitude),only(longitude)),id)*
   f. Na seção **Tamanho e forma**, **Triângulo** é selecionado para **Forma**, e o **Tamanho** é diminuído no controle deslizante de preferência.
   g. Em **Cores e legenda**, **Personalizado** foi selecionado para **Cores**.
   h. A opção **Por dimensão** foi selecionada para colorir o gráfico. A seguinte expressão é inserida: *=pick(aggr(KMeans2D(vDistClusters,only(latitude),only(longitude)),id)+1,'Cluster 1','Cluster 2','Cluster 3','Cluster 4','Cluster 5')*
   i. A dimensão é rotulada *Agrupamentos*.
4. Em **Configurações de mapa**, **Adaptativo** é selecionado para **Projeção**. **Métrica** é selecionado para **Unidades de medida.**

Mapa: Centroides mapeados por agrupamento

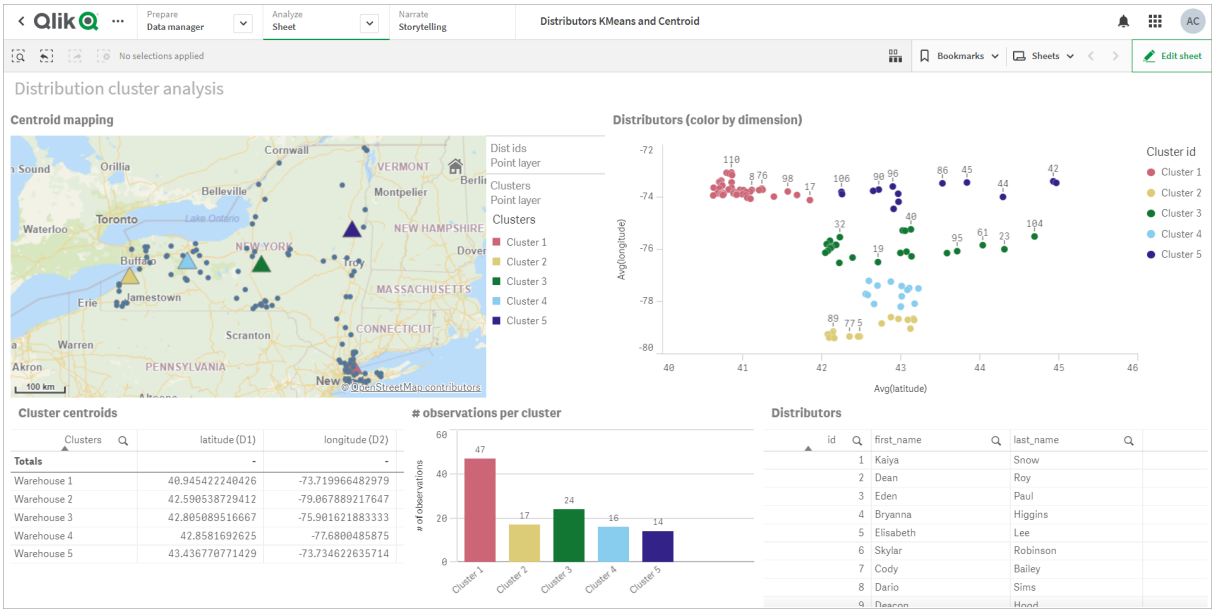


Conclusão

Usando a função KMeans para este cenário do mundo real, os distribuidores foram segmentados em grupos ou agrupamentos semelhantes com base na similaridade; neste caso, proximidade um do outro A função Centroide foi aplicada a esses agrupamentos para identificar cinco coordenadas de mapeamento. Essas

coordenadas fornecem uma localização central inicial para construir ou localizar depósitos. A função do centroide é aplicada ao gráfico de **mapa**, para que os usuários do aplicativo possam visualizar onde os centroides estão localizados em relação aos pontos de dados do agrupamento ao redor. As coordenadas resultantes representam locais de depósito em potencial que poderiam minimizar os custos de entrega para distribuidores no estado de Nova York.

Aplicativo: Exemplo de análise de KMeans e de centroide



Conjunto de dados do Distributor: Carregamento inline para o editor de carregamento de dados no 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 - função de gráfico

O **KMeans2D()** avalia as linhas do gráfico por meio da aplicação do agrupamento de k-means e, para cada linha do gráfico, exibe o ID do agrupamento ao qual esse ponto de dados foi atribuído. As colunas que são usadas pelo algoritmo de agrupamento são determinadas pelos parâmetros coordinate_1 e coordinate_2, respectivamente. Ambas são agregações. O número de agrupamentos criados é determinado pelo parâmetro num_clusters. Os dados podem ser normalizados opcionalmente pelo parâmetro norm.

KMeans2D retorna um valor por ponto de dados. O valor retornado é duplo e é o valor inteiro correspondente ao agrupamento ao qual cada ponto de dados foi atribuído.

Sintaxe:

```
KMeans2D(num_clusters, coordinate_1, coordinate_2 [, norm])
```

Tipo de dados de retorno: dual

Argumentos:

Argumentos

Argumento	Descrição
num_clusters	Inteiro que especifica o número de agrupamentos.
coordinate_1	A agregação que calcula a primeira coordenada, geralmente o eixo X do gráfico de dispersão que pode ser criado a partir do gráfico. O parâmetro adicional, coordinate_2, calcula a segunda coordenada.
norm	O método de normalização opcional aplicado a conjuntos de dados antes do agrupamento KMeans. Valores possíveis: 0 ou "none" para nenhuma normalização 1 ou "zscore" para normalização z-ponto 2 ou "minmax" para normalização mín-máx Se nenhum parâmetro for fornecido ou se o parâmetro fornecido estiver incorreto, nenhuma normalização será aplicada. Z-ponto normaliza os dados com base na média e no desvio padrão do recurso. Z-ponto não garante que cada recurso tenha a mesma escala, mas é uma abordagem melhor que mín-máx ao se lidar com discrepâncias. A normalização mín-máx garante que os recursos tenham a mesma escala, usando os valores mínimo e máximo de cada um e recalculando cada ponto de dados.

Exemplo: Expressão de gráfico
Neste exemplo, criamos um gráfico de dispersão usando o conjunto de dados *Iris* e, em seguida, usamos KMeans para colorir os dados por expressão.

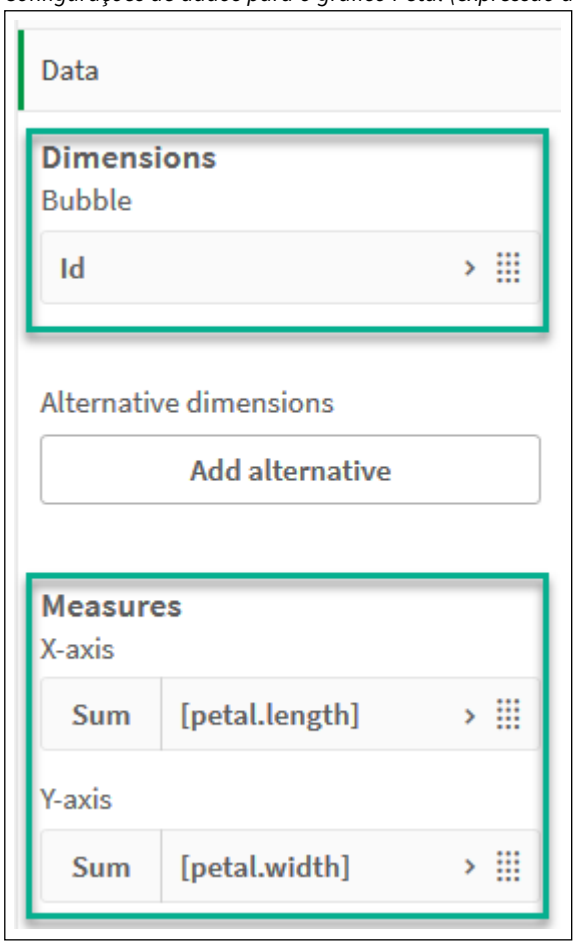
Também criamos uma variável para o argumento *num_clusters* e, em seguida, usamos uma caixa de entrada de variável para alterar o número de agrupamentos.

O conjunto de dados *Iris* está disponível publicamente em uma variedade de formatos. Fornecemos os dados como uma tabela inline para carregar usando o editor de carregamento de dados no Qlik Sense. Observe que adicionamos uma coluna *Id* à tabela de dados para este exemplo.

Depois de carregar os dados no Qlik Sense, faremos o seguinte:

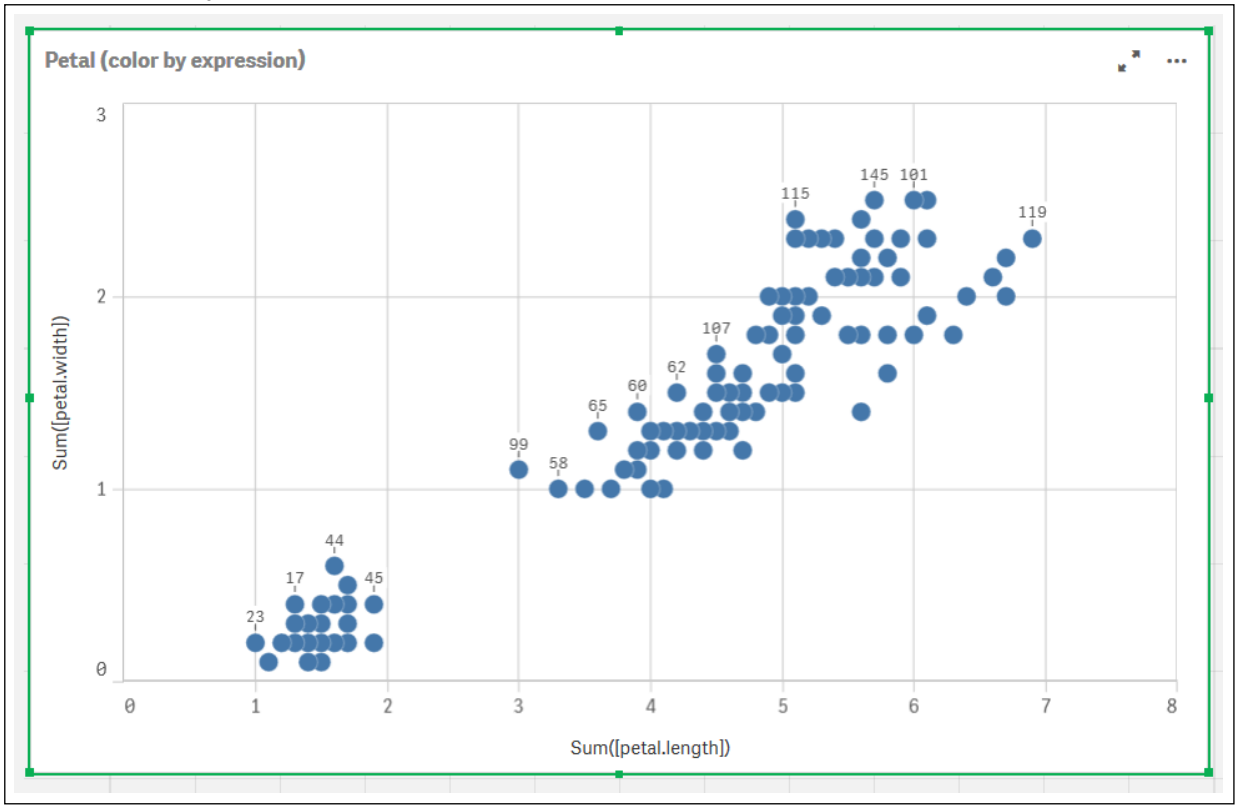
1. Arraste um **Gráfico de dispersão** até uma nova pasta. Especifique o nome *Petal (expressão de cor)* para o gráfico.
2. Crie uma variável para especificar o número de agrupamentos. Para a variável **Nome**, insira *KmeansPetalClusters*. Para a variável **Definição**, insira *=2*.
3. Configure **Dados** para o gráfico:
    i. Em **Dimensões**, escolha *id* para o campo de **Bolha**. Insira ID do Agrupamento para o Rótulo.
    ii. Em **Medidas**, escolha *Sum([petal.length])* para a expressão do **Eixo X**.
    iii. Em **Medidas**, escolha *Sum([petal.width])* para a expressão do **Eixo Y**.

Configurações de dados para o gráfico Petal (expressão de cor)



Os pontos de dados são plotados no gráfico.

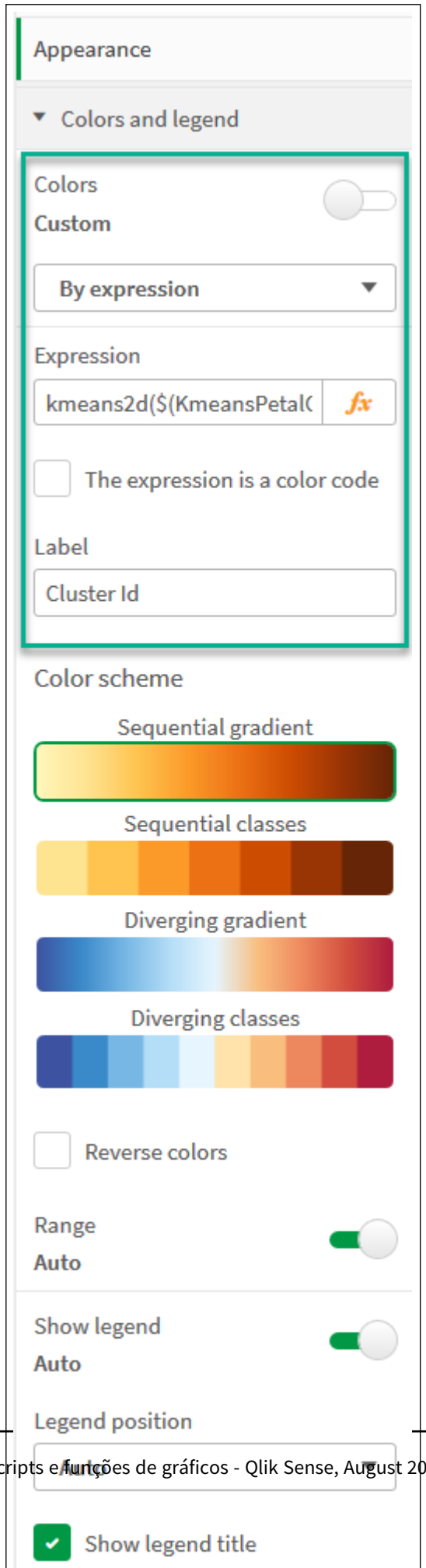
Pontos de dados no gráfico Petal (expressão de cor)



4. Configure a **Aparência** do gráfico:
    i. Em **Cores e legendas**, escolha **Personalizado** para **Cores**.
    ii. Escolha colorir o gráfico **Por expressão**.
    iii. Insira o seguinte para **Expressão**: *kmeans2d($(KmeansPetalClusters), Sum([petal.length]), Sum([petal.width]))*
    Observe que *KmeansPetalClusters* é a variável que definimos como *2*.
    Como alternativa, insira o seguinte: *kmeans2d(2, Sum([petal.length]), Sum([petal.width]))*
    iv. Desmarque a caixa de seleção para **A expressão é um código de cor**.

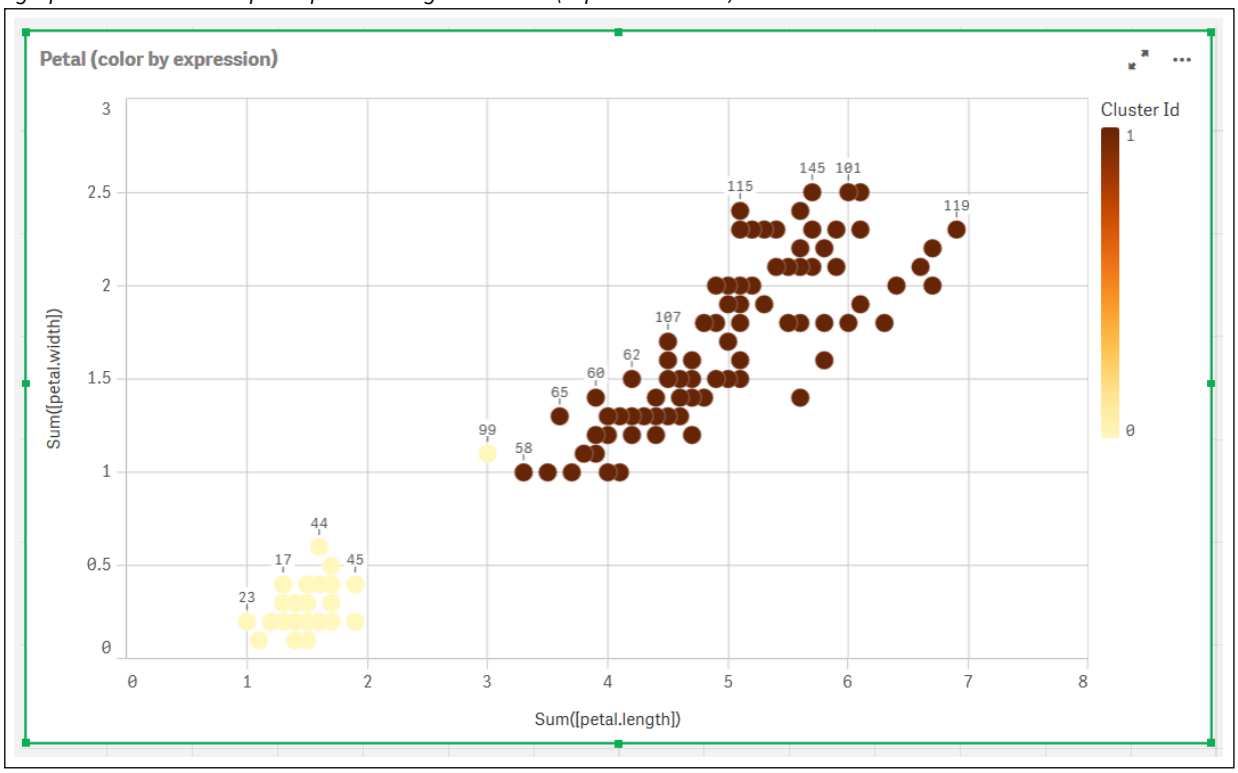
v. Insira o seguinte para **Rótulo**: *ID do agrupamento*

Configurações de aparência para o gráfico Petal (expressão de cor)


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
Show legend title

Os dois agrupamentos no gráfico são coloridos pela expressão KMeans.

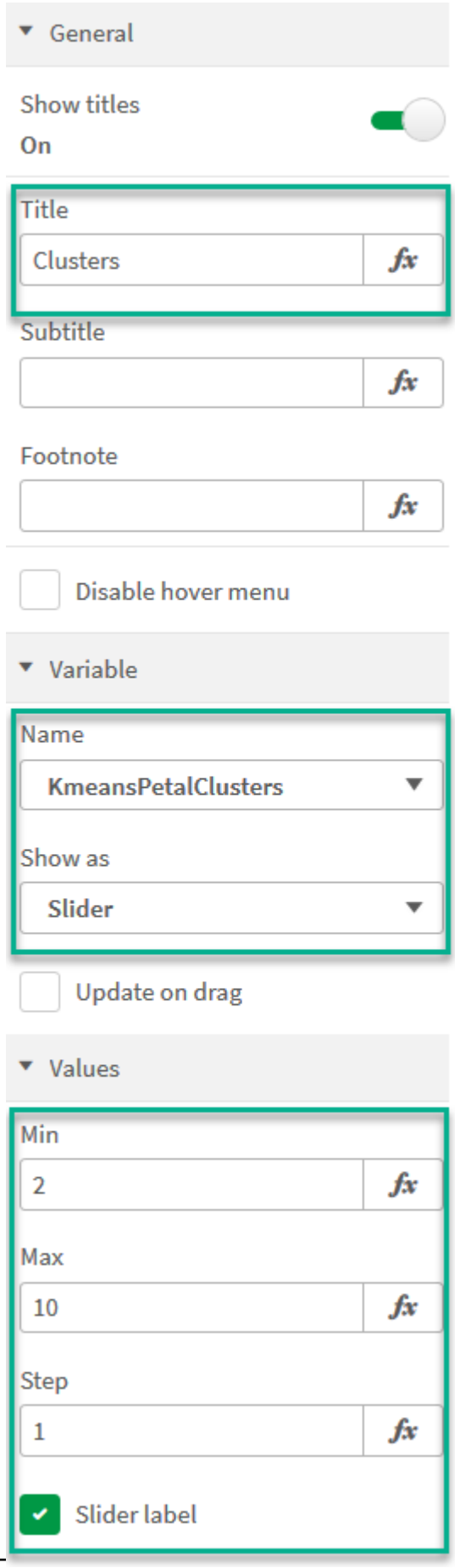
Agrupamentos coloridos por expressão no gráfico Petal (expressão de cor)



5. Adicione uma caixa de **Entrada variável** para o número de agrupamentos.
   i. Em **Objetos personalizados** no painel **Ativos**, escolha **Pacote Dashboard da Qlik**. Se não tivéssemos acesso ao pacote dashboard, ainda poderíamos alterar o número de agrupamentos usando a variável que criamos ou diretamente como um inteiro na expressão.
   ii. Arraste uma caixa de **Entrada variável** até a pasta.
   iii. Em **Aparência**, clique em **Geral**.
   iv. Insira o seguinte para **Título**: *Agrupamentos*
   v. Clique em **Variável**.
   vi. Escolha a seguinte variável para **Nome**: *KmeansPetalClusters*.
   vii. Escolha **Controle Deslizante** para **Mostrar como**.

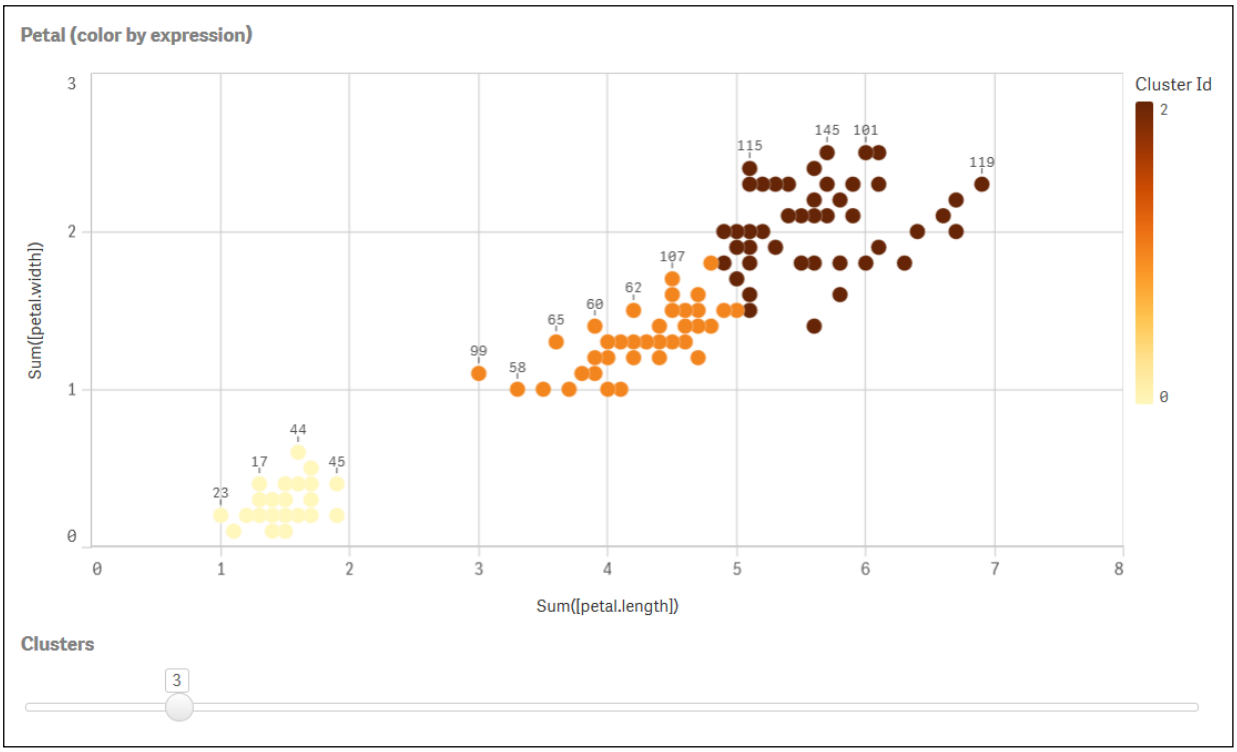
viii. Escolha **Valores** e defina as configurações conforme necessário,

Aparência da caixa de entrada da variável Agrupamentos


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

Quando terminamos de editar, podemos alterar o número de agrupamentos usando o controle deslizante na caixa de entrada variável *Agrupamentos*.

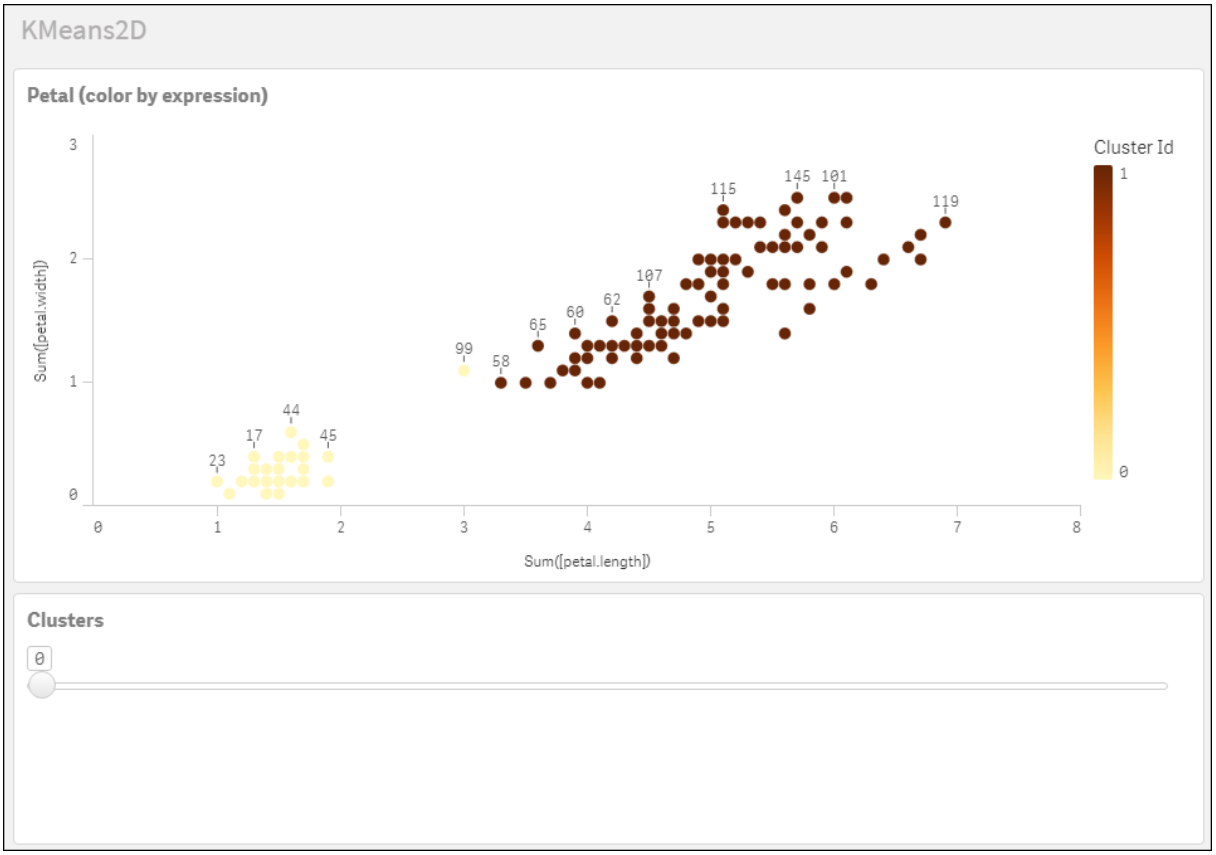
Agrupamentos coloridos por expressão no gráfico Petal (expressão de cor)



Agrupamento automático

Funções **KMeans** oferecem suporte para agrupamento automático usando um método chamado de diferença de profundidade (DeD). Quando um usuário define 0 para o número de agrupamentos, um número ideal de agrupamentos para esse conjunto de dados é determinado. Observe que, embora um número inteiro para o número de agrupamentos (*k*) não seja retornado explicitamente, ele é calculado dentro do algoritmo KMeans. Por exemplo, se 0 for especificado na função para o valor de*KmeansPetalClusters* ou definido por meio de uma caixa de entrada de variável, atribuições de agrupamentos serão calculadas automaticamente para o conjunto de dados com base em um número ideal de agrupamentos.

O método de diferença de profundidade KMeans determina o número ideal de agrupamentos quando (k) está definido como 0



Conjunto de dados Iris: Carregamento inline para o editor de carregamento de dados no 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 - função de gráfico

O **KMeansND()** avalia as linhas do gráfico por meio da aplicação do agrupamento de k-means e, para cada linha do gráfico, exibe o ID do agrupamento ao qual esse ponto de dados foi atribuído. As colunas que são usadas pelo algoritmo de agrupamento são determinadas pelos parâmetros coordinate_1 e coordinate_2, etc., até n colunas. Todas são agregações. O número de agrupamentos criados é determinado pelo parâmetro num_clusters.

KMeansND retorna um valor por ponto de dados. O valor retornado é duplo e é o valor inteiro correspondente ao agrupamento ao qual cada ponto de dados foi atribuído.

Sintaxe:

```
KMeansND(num_clusters, num_iter, coordinate_1, coordinate_2 [,coordinate_3 [,...]])
```

Tipo de dados de retorno: dual

Argumentos:

Argumentos

Argumento	Descrição
num_clusters	Inteiro que especifica o número de agrupamentos.
num_iter	O número de iterações de agrupamentos com centros de agrupamentos reinicializados.
coordinate_1	A agregação que calcula a primeira coordenada, geralmente o eixo X (de um gráfico de dispersão que pode ser criado a partir do gráfico). Os parâmetros adicionais calculam a segunda, a terceira e a quarta coordenadas, etc.

Exemplo: Expressão de gráfico
Neste exemplo, criamos um gráfico de dispersão usando o conjunto de dados *Iris* e, em seguida, usamos KMeans para colorir os dados por expressão.

Também criamos uma variável para o argumento *num_clusters* e, em seguida, usamos uma caixa de entrada de variável para alterar o número de agrupamentos.

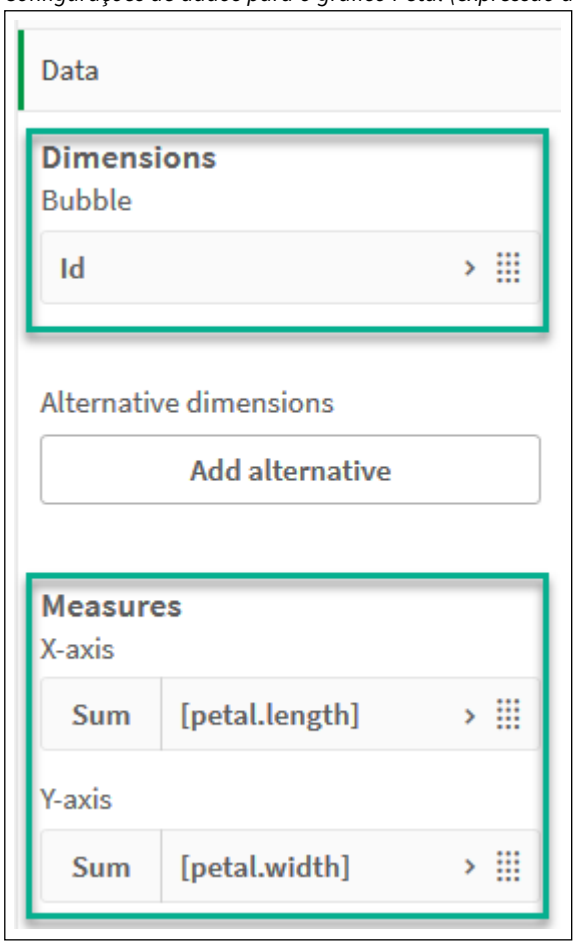
Também criamos uma variável para o argumento *num_iter* e, em seguida, usamos uma segunda caixa de entrada de variável para alterar o número de iterações.

O conjunto de dados *Iris* está disponível publicamente em uma variedade de formatos. Fornecemos os dados como uma tabela inline para carregar usando o editor de carregamento de dados no Qlik Sense. Observe que adicionamos uma coluna *Id* à tabela de dados para este exemplo.

Depois de carregar os dados no Qlik Sense, faremos o seguinte:

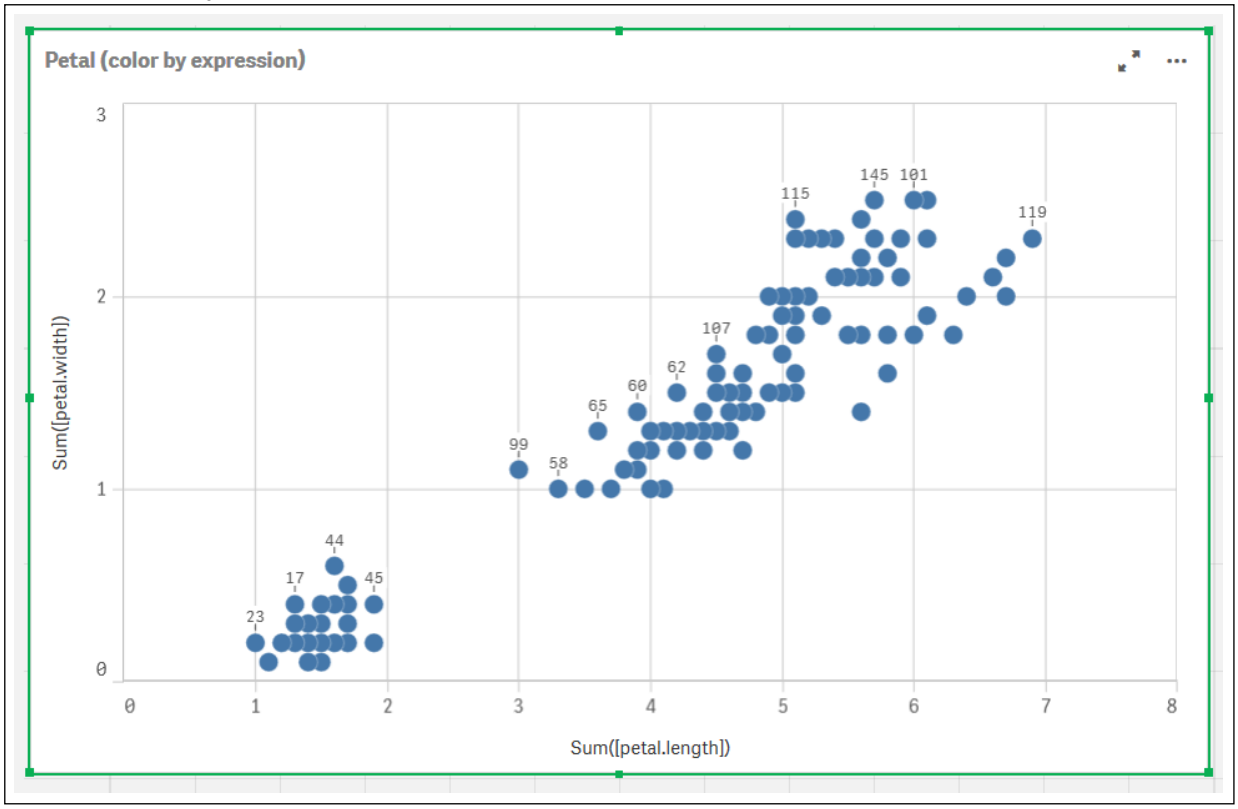
1. Arraste um **Gráfico de dispersão** até uma nova pasta. Especifique o nome *Petal (expressão de cor)* para o gráfico.
2. Crie uma variável para especificar o número de agrupamentos. Para a variável **Nome**, insira *KmeansPetalClusters*. Para a variável **Definição**, insira *=2*.
3. Crie uma variável para especificar o número de iterações. Para a variável **Nome**, insira *KmeansNumberIterations*. Para a variável **Definição**, insira *=1*.
4. Configure **Dados** para o gráfico:
    i. Em **Dimensões**, escolha *id* para o campo de **Bolha**. Insira ID do Agrupamento para o Rótulo.
    ii. Em **Medidas**, escolha *Sum([petal.length])* para a expressão do **Eixo X**.
    iii. Em **Medidas**, escolha *Sum([petal.width])* para a expressão do **Eixo Y**.

Configurações de dados para o gráfico Petal (expressão de cor)



Os pontos de dados são plotados no gráfico.

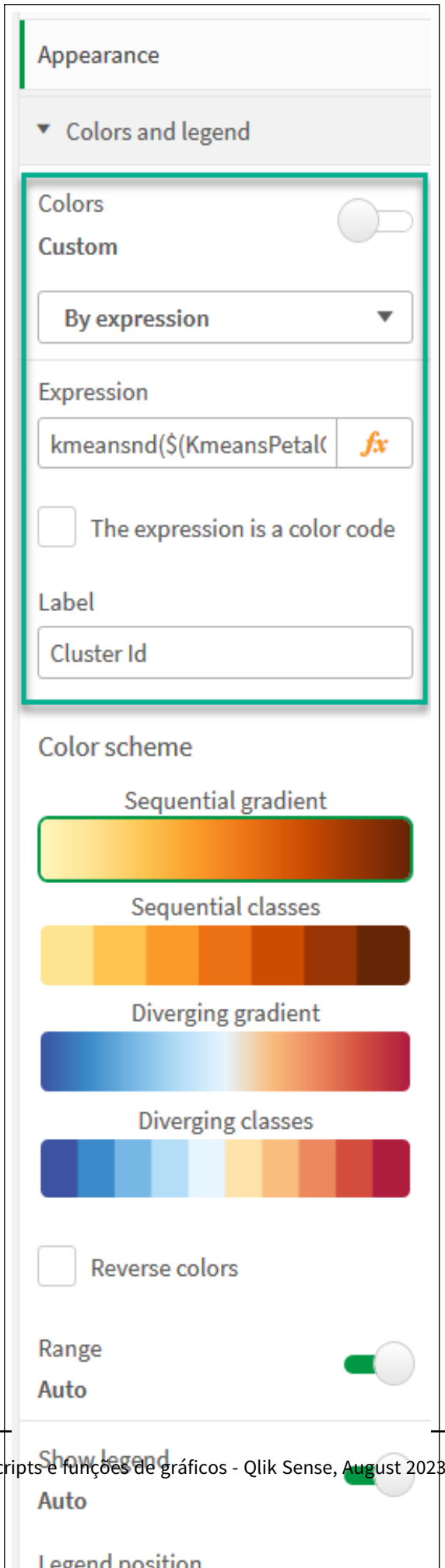
Pontos de dados no gráfico Petal (expressão de cor)



5. Configure a **Aparência** do gráfico:
    i. Em **Cores e legendas**, escolha **Personalizado** para **Cores**.
    ii. Escolha colorir o gráfico **Por expressão**.
    iii. Insira o seguinte para **Expressão**: *kmeansnd ($(KmeansPetalClusters),$(KmeansNumberIterations), Sum([petal.length]), Sum ([petal.width]),Sum([sepal.length]), Sum([sepal.width]))*
    Observe que *KmeansPetalClusters* é a variável que definimos como *2*. *KmeansNumberIterations* é a variável que definimos como *1*.
    Como alternativa, insira o seguinte: *kmeansnd(2, 2, Sum([petal.length]), Sum([petal.width]),Sum ([sepal.length]), Sum([sepal.width]))*
    iv. Desmarque a caixa de seleção para **A expressão é um código de cor**.

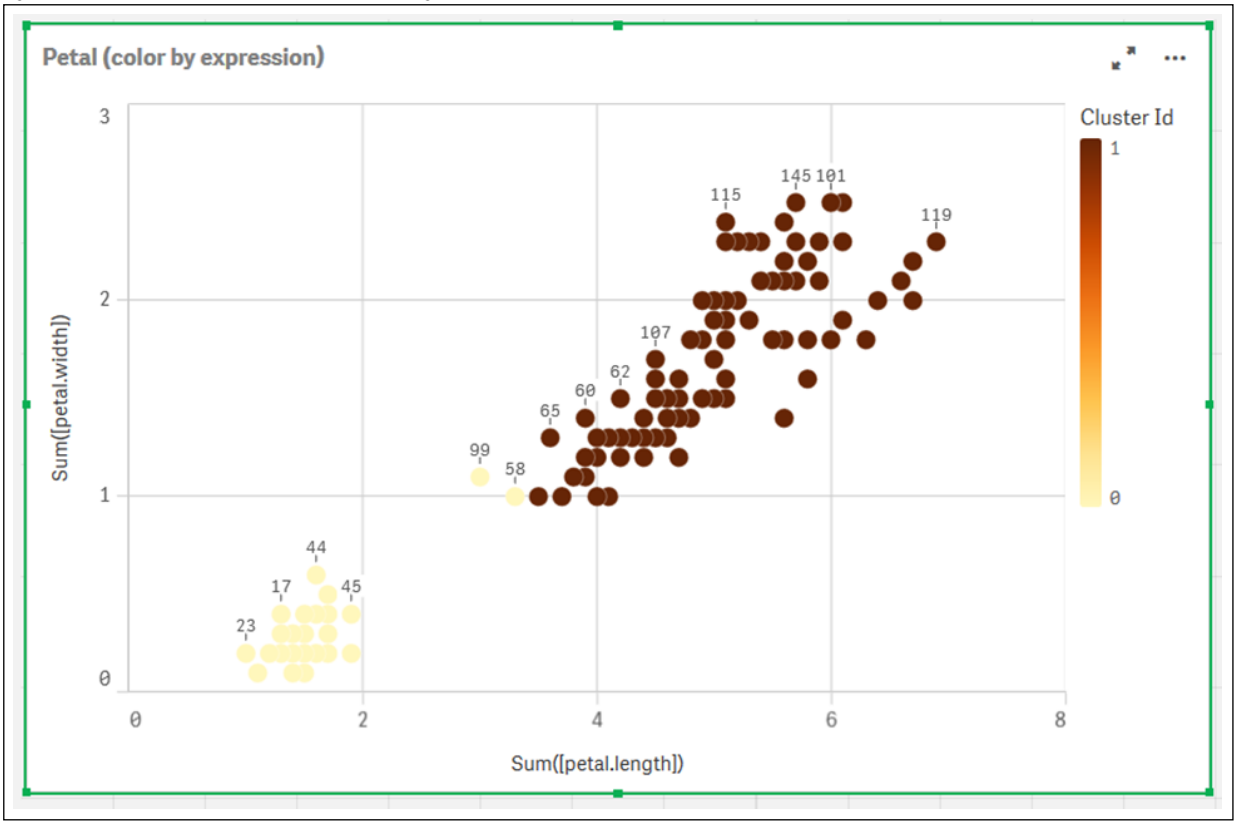
v. Insira o seguinte para **Rótulo**: *ID do agrupamento*

Configurações de aparência para o gráfico Petal (expressão de cor)


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

Os dois agrupamentos no gráfico são coloridos pela expressão KMeans.

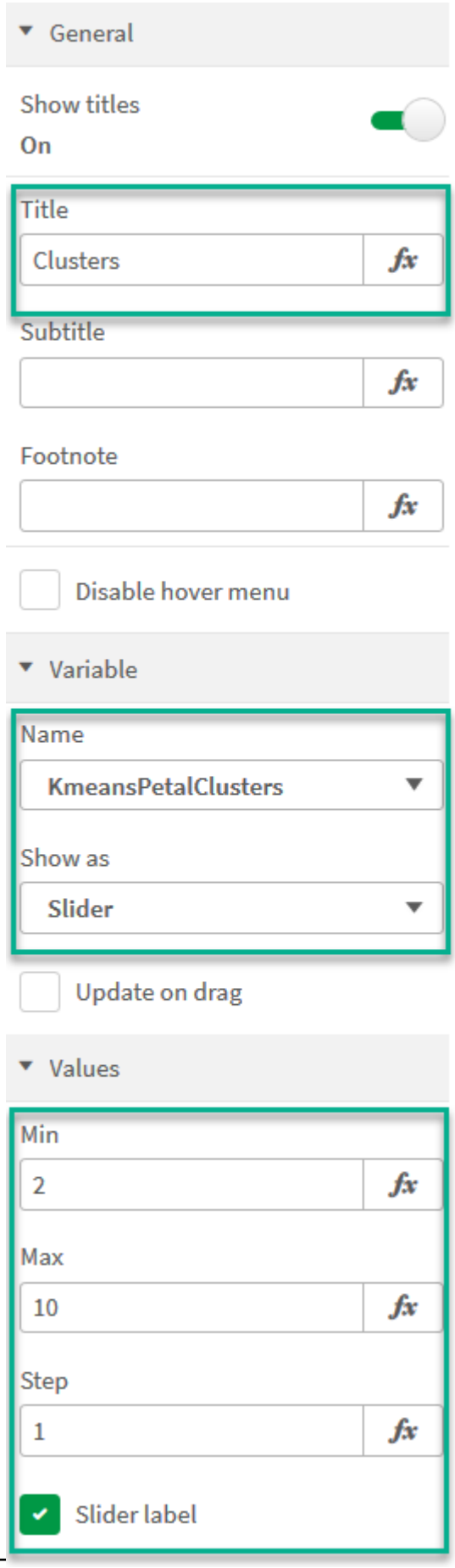
Agrupamentos coloridos por expressão no gráfico Petal (expressão de cor)



6. Adicione uma caixa de **Entrada variável** para o número de agrupamentos.
    i. Em **Objetos personalizados** no painel **Ativos**, escolha **Pacote Dashboard da Qlik**. Se não tivéssemos acesso ao pacote dashboard, ainda poderíamos alterar o número de agrupamentos usando a variável que criamos ou diretamente como um inteiro na expressão.
    ii. Arraste uma caixa de **Entrada variável** até a pasta.
    iii. Em **Aparência**, clique em **Geral**.
    iv. Insira o seguinte para **Título**: *Agrupamentos*
    v. Clique em **Variável**.
    vi. Escolha a seguinte variável para **Nome**: *KmeansPetalClusters*.
    vii. Escolha **Controle Deslizante** para **Mostrar como**.

viii. Escolha **Valores** e defina as configurações conforme necessário,

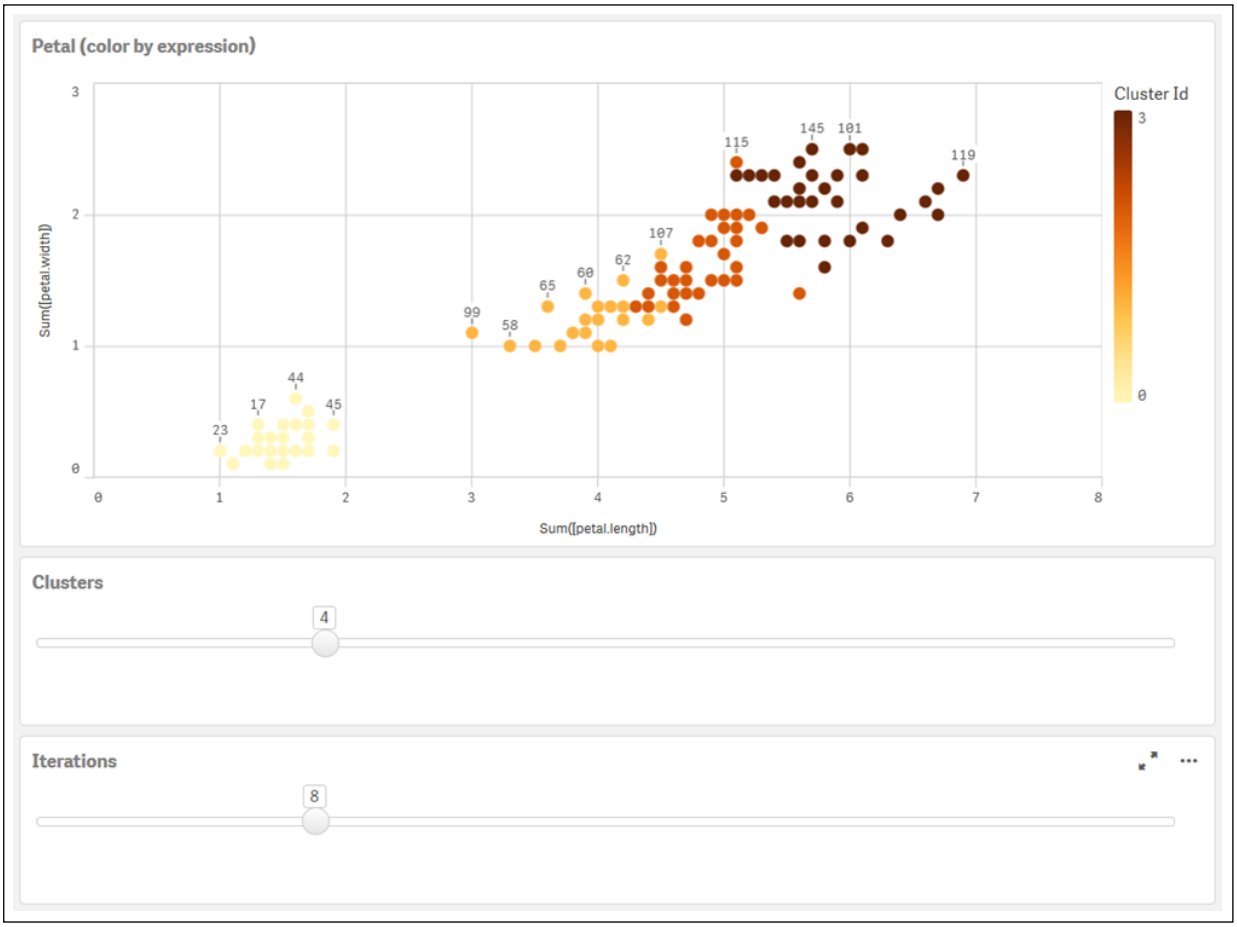
Aparência da caixa de entrada da variável Agrupamentos


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. Adicione uma caixa de **Entrada variável** para o número de iterações.
    i. Arraste uma caixa de **Entrada variável** até a pasta.
    ii. Em **Aparência**, escolha **Geral**.
    iii. Insira o seguinte para **Título**: *Iterações*
    iv. Em **Aparência**, escolha **Variável**.
    v. Escolha a seguinte variável em **Nome**: *KmeansNumberIterations*.
    vi. Defina as configurações adicionais conforme necessário,

    Agora, podemos alterar o número de agrupamentos e iterações usando os controles deslizantes nas caixas de entrada de variáveis.

    Agrupamentos coloridos por expressão no gráfico Petal (expressão de cor)

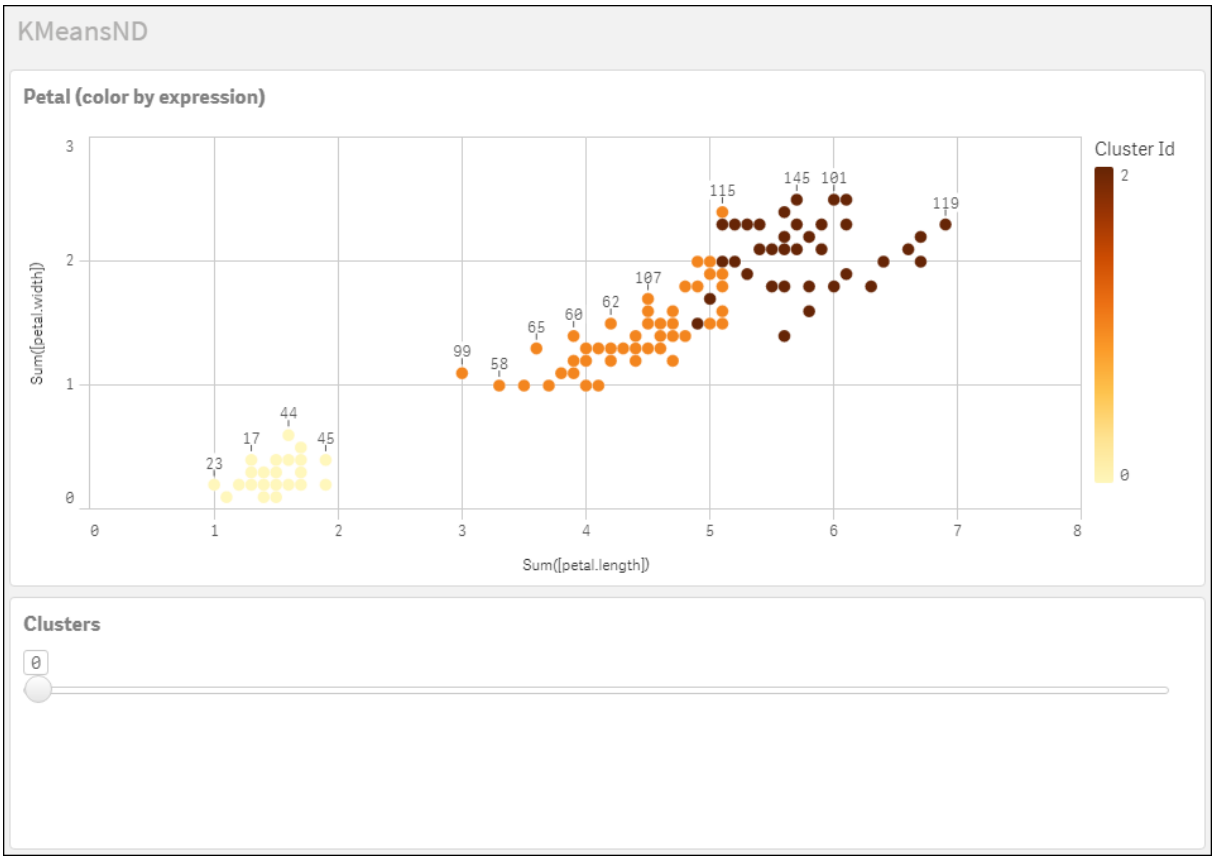


Agrupamento automático

Funções **KMeans** oferecem suporte para agrupamento automático usando um método chamado de diferença de profundidade (DeD). Quando um usuário define 0 para o número de agrupamentos, um número ideal de agrupamentos para esse conjunto de dados é determinado. Observe que, embora um número inteiro para o número de agrupamentos (*k*) não seja retornado explicitamente, ele é calculado dentro do algoritmo KMeans. Por exemplo, se 0 for especificado na função para o valor de*KmeansPetalClusters* ou definido por meio de

uma caixa de entrada de variável, atribuições de agrupamentos serão calculadas automaticamente para o conjunto de dados com base em um número ideal de agrupamentos. Dado o conjunto de dados de Iris, se 0 for selecionado para o número de agrupamentos, o algoritmo determinará (agrupará automaticamente) um número ideal de agrupamentos (3) para esse conjunto de dados.

O método de diferença de profundidade KMeans determina o número ideal de agrupamentos quando (k) está definido como 0.



Conjunto de dados Iris: Carregamento inline para o editor de carregamento de dados no 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 - função de gráfico

O **KMeansCentroid2D()** avalia as linhas do gráfico por meio da aplicação do agrupamento de k-means e, para cada linha do gráfico, exibe a coordenada desejada do agrupamento ao qual esse ponto de dados foi atribuído. As colunas que são usadas pelo algoritmo de agrupamento são determinadas pelos parâmetros coordinate_1 e coordinate_2, respectivamente. Ambas são agregações. O número de agrupamentos criados é determinado pelo parâmetro num_clusters. Os dados podem ser normalizados opcionalmente pelo parâmetro norm.

KMeansCentroid2D retorna um valor por ponto de dados. O valor retornado é duplo e é uma das coordenadas da posição correspondente ao centro do agrupamento ao qual o ponto de dados foi atribuído.

Sintaxe:

```
KMeansCentroid2D(num_clusters, coordinate_no, coordinate_1, coordinate_2 [, norm])
```

Tipo de dados de retorno: dual

Argumentos:

Argumentos

Argumento	Descrição
num_clusters	Inteiro que especifica o número de agrupamentos.
coordinate_no	O número da coordenada desejada dos centroides (correspondendo, por exemplo, ao eixo X, Y ou Z).
coordinate_1	A agregação que calcula a primeira coordenada, geralmente o eixo X do gráfico de dispersão que pode ser criado a partir do gráfico. O parâmetro adicional, coordinate_2, calcula a segunda coordenada.
norm	O método de normalização opcional aplicado a conjuntos de dados antes do agrupamento KMeans. Valores possíveis: 0 ou "none" para nenhuma normalização 1 ou "zscore" para normalização z-ponto 2 ou "minmax" para normalização mín-máx Se nenhum parâmetro for fornecido ou se o parâmetro fornecido estiver incorreto, nenhuma normalização será aplicada. Z-ponto normaliza os dados com base na média e no desvio padrão do recurso. Z-ponto não garante que cada recurso tenha a mesma escala, mas é uma abordagem melhor que mín-máx ao se lidar com discrepâncias. A normalização mín-máx garante que os recursos tenham a mesma escala, usando os valores mínimo e máximo de cada um e recalculando cada ponto de dados.

Agrupamento automático

Funções **KMeans** oferecem suporte para agrupamento automático usando um método chamado de diferença de profundidade (DeD). Quando um usuário define 0 para o número de agrupamentos, um número ideal de agrupamentos para esse conjunto de dados é determinado. Observe que, embora um número inteiro para o número de agrupamentos (*k*) não seja retornado explicitamente, ele é calculado dentro do algoritmo KMeans. Por exemplo, se 0 for especificado na função para o valor de*KmeansPetalClusters* ou definido por meio de uma caixa de entrada de variável, atribuições de agrupamentos serão calculadas automaticamente para o conjunto de dados com base em um número ideal de agrupamentos.

KMeansCentroidND - função de gráfico

O **KMeansCentroidND()** avalia as linhas do gráfico por meio da aplicação do agrupamento de k-means e, para cada linha do gráfico, exibe a coordenada desejada do agrupamento ao qual esse ponto de dados foi atribuído. As colunas que são usadas pelo algoritmo de agrupamento são determinadas pelos parâmetros coordinate_1, coordinate_2, etc., até n colunas. Todas são agregações. O número de agrupamentos criados é determinado pelo parâmetro num_clusters.

KMeansCentroidND retorna um valor por linha. O valor retornado é duplo e é uma das coordenadas da posição correspondente ao centro do agrupamento ao qual o ponto de dados foi atribuído.

Sintaxe:

```
KMeansCentroidND((num_clusters, num_iter, coordinate_no, coordinate_1,
coordinate_2 [,coordinate_3 [, ...]])
```

Tipo de dados de retorno: dual

Argumentos:

Argumentos

Argumento	Descrição
num_clusters	Inteiro que especifica o número de agrupamentos.
num_iter	O número de iterações de agrupamentos com centros de agrupamentos reinicializados.
coordinate_no	O número da coordenada desejada dos centroides (correspondendo, por exemplo, ao eixo X, Y ou Z).
coordinate_1	A agregação que calcula a primeira coordenada, geralmente o eixo X (de um gráfico de dispersão que pode ser criado a partir do gráfico). Os parâmetros adicionais calculam a segunda, a terceira e a quarta coordenadas, etc.

Agrupamento automático

Funções **KMeans** oferecem suporte para agrupamento automático usando um método chamado de diferença de profundidade (DeD). Quando um usuário define 0 para o número de agrupamentos, um número ideal de agrupamentos para esse conjunto de dados é determinado. Observe que, embora um número inteiro para o número de agrupamentos (*k*) não seja retornado explicitamente, ele é calculado dentro do algoritmo KMeans. Por exemplo, se 0 for especificado na função para o valor de*KmeansPetalClusters* ou definido por meio de uma caixa de entrada de variável, atribuições de agrupamentos serão calculadas automaticamente para o conjunto de dados com base em um número ideal de agrupamentos.

STL_Trend - função de gráfico

STL_Trend é uma função de decomposição de séries temporais. Junto com **STL_Seasonal** e **STL_Residual**, essa função é usada para decompor uma série temporal em componentes sazonais, de tendências e residuais. No contexto do algoritmo STL, a decomposição de séries temporais é usada para identificar tanto um padrão sazonal recorrente quanto uma tendência geral, considerando uma métrica de entrada e outros parâmetros. A função **STL_Trend** identificará uma tendência geral, independente dos padrões ou ciclos sazonais, a partir de dados de séries temporais.

As três funções STL estão relacionadas à métrica de entrada por meio de uma soma simples:

STL_Trend + **STL_Seasonal** + **STL_Residual** = métrica de entrada

A STL (decomposição sazonal e de tendências usando Loess) emprega técnicas de suavização de dados e, por meio de seus parâmetros de entrada, permite ao usuário ajustar a periodicidade dos cálculos que ela realiza. Essa periodicidade determina como a dimensão temporal da métrica de entrada (uma medida) é segmentada na análise.

No mínimo, **STL_Trend** usa uma métrica de entrada (`target_measure`) e um valor inteiro para sua `period_int`, retornando um valor de ponto flutuante. A métrica de entrada terá a forma de uma agregação que varia ao longo da dimensão de tempo. Opcionalmente, você pode incluir valores para `seasonal_smoother` e `trend_smoother` para ajustar o algoritmo de suavização.

Sintaxe:

```
STL_Trend(target_measure, period_int [,seasonal_smoother [,trend_smoother]])
```

Tipo de dados de retorno: dual

Argumentos

Argumento	Descrição
target_measure	A medida a ser decomposta em componentes sazonais e de tendência. Esta deve ser uma medida como Sum(Sales) ou Sum(Passengers) que varia ao longo da dimensão do tempo. Isso não deve ser um valor constante.
period_int	A periodicidade do conjunto de dados. Esse parâmetro é um valor inteiro que representa o número de etapas discretas que compõem um período, ou ciclo sazonal, do sinal. Por exemplo, se a série temporal for segmentada em uma seção para cada trimestre do ano, você deverá definir **period_int** com um valor de 4 para especificar a periodicidade como Ano.

Argumento	Descrição
seasonal_smoother	Duração do suavizador sazonal. Deve ser um número inteiro ímpar. O suavizador sazonal usa dados para uma fase específica da variação sazonal ao longo de vários períodos. Uma etapa discreta da dimensão de tempo é usada de cada período. O suavizador sazonal indica o número de períodos usados para suavização. Por exemplo, se a dimensão de tempo for segmentada por mês e o período for Ano (12), o componente sazonal será calculado de forma que cada mês específico de cada ano seja calculado a partir dos dados do mesmo mês, tanto nesse ano quanto nos anos adjacentes. O valor de **seasonal_smoother** é o número de anos usados para suavização.
trend_smoother	Duração do suavizador de tendência. Deve ser um número inteiro ímpar. O suavizador de tendência usa a mesma escala de tempo do parâmetro **period_int**, e seu valor é o número de grânulos usados para suavização. Por exemplo, se uma série temporal for segmentada por mês, o suavizador de tendência será o número de meses usados para suavização.

A função de gráfico **STL_Trend** é frequentemente usada em combinação com as seguintes funções:

Funções relacionadas

Função	Interação
STL_Seasonal - função de gráfico (page 1430)	Essa é a função usada para calcular o componente sazonal de uma série temporal.
STL_Residual - função de gráfico (page 1432)	Ao dividir uma métrica de entrada em um componente sazonal e um componente de tendência, parte da variação da medida não caberá em nenhum dos dois componentes principais. A função **STL_Residual** calcula essa parte da decomposição.

Para ver um tutorial com um exemplo completo que mostra como usar essa função, consulte *Tutorial - Decomposição de séries temporais no Qlik Sense (page 1434)*.

STL_Seasonal - função de gráfico

STL_Seasonal é uma função de decomposição de séries temporais. Junto com **STL_Trend** e **STL_Residual**, essa função é usada para decompor uma série temporal em componentes sazonais, de tendências e residuais. No contexto do algoritmo STL, a decomposição de séries temporais é usada para identificar tanto um padrão sazonal recorrente quanto uma tendência geral, considerando uma métrica de entrada e outros parâmetros. A função **STL_Seasonal** pode identificar um padrão sazonal em uma série temporal, separando-o da tendência geral exibida pelos dados.

As três funções STL estão relacionadas à métrica de entrada por meio de uma soma simples:

STL_Trend + **STL_Seasonal** + **STL_Residual** = métrica de entrada

A STL (decomposição sazonal e de tendências usando Loess) emprega técnicas de suavização de dados e, por meio de seus parâmetros de entrada, permite ao usuário ajustar a periodicidade dos cálculos que ela realiza. Essa periodicidade determina como a dimensão temporal da métrica de entrada (uma medida) é segmentada na análise.

No mínimo, **STL_Seasonal** usa uma métrica de entrada (`target_measure`) e um valor inteiro para sua `period_int`, retornando um valor de ponto flutuante. A métrica de entrada terá a forma de uma agregação que varia ao longo da dimensão de tempo. Opcionalmente, você pode incluir valores para `seasonal_smoother` e `trend_smoother` para ajustar o algoritmo de suavização.

Sintaxe:

```
STL_Seasonal(target_measure, period_int [,seasonal_smoother [,trend_
smoother]])
```

Tipo de dados de retorno: dual

Argumentos

Argumento	Descrição
target_measure	A medida a ser decomposta em componentes sazonais e de tendência. Esta deve ser uma medida como Sum(Sales) ou Sum(Passengers) que varia ao longo da dimensão do tempo. Isso não deve ser um valor constante.
period_int	A periodicidade do conjunto de dados. Esse parâmetro é um valor inteiro que representa o número de etapas discretas que compõem um período, ou ciclo sazonal, do sinal. Por exemplo, se a série temporal for segmentada em uma seção para cada trimestre do ano, você deverá definir **period_int** com um valor de 4 para especificar a periodicidade como Ano.

Argumento	Descrição
seasonal_ smoother	Duração do suavizador sazonal. Deve ser um número inteiro ímpar. O suavizador sazonal usa dados para uma fase específica da variação sazonal ao longo de vários períodos. Uma etapa discreta da dimensão de tempo é usada de cada período. O suavizador sazonal indica o número de períodos usados para suavização. Por exemplo, se a dimensão de tempo for segmentada por mês e o período for Ano (12), o componente sazonal será calculado de forma que cada mês específico de cada ano seja calculado a partir dos dados do mesmo mês, tanto nesse ano quanto nos anos adjacentes. O valor de **seasonal_smoother** é o número de anos usados para suavização.
trend_ smoother	Duração do suavizador de tendência. Deve ser um número inteiro ímpar. O suavizador de tendência usa a mesma escala de tempo do parâmetro **period_int**, e seu valor é o número de grânulos usados para suavização. Por exemplo, se uma série temporal for segmentada por mês, o suavizador de tendência será o número de meses usados para suavização.

A função de gráfico **STL_Seasonal** é frequentemente usada em combinação com as seguintes funções:

Funções relacionadas

Função	Interação
STL_Trend - função de gráfico (page 1428)	Essa é a função usada para calcular o componente de tendência de uma série temporal.
STL_Residual - função de gráfico (page 1432)	Ao dividir uma métrica de entrada em um componente sazonal e um componente de tendência, parte da variação da medida não caberá em nenhum dos dois componentes principais. A função **STL_ Residual** calcula essa parte da decomposição.

Para ver um tutorial com um exemplo completo que mostra como usar essa função, consulte *Tutorial - Decomposição de séries temporais no Qlik Sense (page 1434)*.

STL_Residual - função de gráfico

STL_Residual é uma função de decomposição de séries temporais. Juntamente com **STL_Seasonal** e **STL_Trend**, essa função é usada para decompor uma série temporal em componentes sazonais, de tendência e residuais. No contexto do algoritmo STL, a decomposição de séries temporais é usada para identificar tanto um padrão sazonal recorrente quanto uma tendência geral, considerando uma métrica de entrada e outros parâmetros. Ao realizar essa operação, parte da variação na métrica de entrada não se encaixará no componente sazonal nem no componente de tendências e será definida como o componente residual. A função de gráfico **STL_Residual** captura essa parte do cálculo.

As três funções STL estão relacionadas à métrica de entrada por meio de uma soma simples:

STL_Trend + **STL_Seasonal** + **STL_Residual** = métrica de entrada

A STL (decomposição sazonal e de tendências usando Loess) emprega técnicas de suavização de dados e, por meio de seus parâmetros de entrada, permite ao usuário ajustar a periodicidade dos cálculos que ela realiza. Essa periodicidade determina como a dimensão temporal da métrica de entrada (uma medida) é segmentada na análise.

Como a decomposição da série temporal busca principalmente a sazonalidade e variações gerais nos dados, as informações no componente residual são consideradas as menos significativas dos três componentes. No entanto, um componente residual distorcido ou periódico pode ajudar a identificar problemas no cálculo, como configurações de periodicidade incorretas.

No mínimo, **STL_Residual** usa uma métrica de entrada (`target_measure`) e um valor inteiro para sua `period_int`, retornando um valor de ponto flutuante. A métrica de entrada terá a forma de uma agregação que varia ao longo da dimensão de tempo. Opcionalmente, você pode incluir valores para `seasonal_smoother` e `trend_smoother` para ajustar o algoritmo de suavização.

Sintaxe:

```
STL_Residual(target_measure, period_int [,seasonal_smoother [,trend_smoother]])
```

Tipo de dados de retorno: dual

Argumentos

Argumento	Descrição
target_measure	A medida a ser decomposta em componentes sazonais e de tendência. Esta deve ser uma medida como Sum(Sales) ou Sum(Passengers) que varia ao longo da dimensão do tempo. Isso não deve ser um valor constante.

Argumento	Descrição
period_int	A periodicidade do conjunto de dados. Esse parâmetro é um valor inteiro que representa o número de etapas discretas que compõem um período, ou ciclo sazonal, do sinal. Por exemplo, se a série temporal for segmentada em uma seção para cada trimestre do ano, você deverá definir **period_int** com um valor de 4 para especificar a periodicidade como Ano.
seasonal_ smoother	Duração do suavizador sazonal. Deve ser um número inteiro ímpar. O suavizador sazonal usa dados para uma fase específica da variação sazonal ao longo de vários períodos. Uma etapa discreta da dimensão de tempo é usada de cada período. O suavizador sazonal indica o número de períodos usados para suavização. Por exemplo, se a dimensão de tempo for segmentada por mês e o período for Ano (12), o componente sazonal será calculado de forma que cada mês específico de cada ano seja calculado a partir dos dados do mesmo mês, tanto nesse ano quanto nos anos adjacentes. O valor de **seasonal_smoother** é o número de anos usados para suavização.
trend_ smoother	Duração do suavizador de tendência. Deve ser um número inteiro ímpar. O suavizador de tendência usa a mesma escala de tempo do parâmetro **period_int**, e seu valor é o número de grânulos usados para suavização. Por exemplo, se uma série temporal for segmentada por mês, o suavizador de tendência será o número de meses usados para suavização.

A função de gráfico **STL_Residual** é frequentemente usada em combinação com as seguintes funções:

Funções relacionadas

Função	Interação
STL_Seasonal - função de gráfico (page 1430)	Essa é a função usada para calcular o componente sazonal de uma série temporal.
STL_Trend - função de gráfico (page 1428)	Essa é a função usada para calcular o componente de tendência de uma série temporal.

Para ver um tutorial com um exemplo completo que mostra como usar essa função, consulte *Tutorial - Decomposição de séries temporais no Qlik Sense (page 1434)*.

Tutorial - Decomposição de séries temporais no Qlik Sense

Este tutorial demonstra o uso de três funções de gráfico para decompor uma série temporal usando o algoritmo STL.

Este tutorial usa dados de séries temporais do número de passageiros que usam uma companhia aérea por mês para demonstrar a funcionalidade do algoritmo STL. As funções de gráfico **STL_Trend**, **STL_Seasonal** e **STL_Residual** serão usadas para criar as visualizações. Para obter mais informações sobre a decomposição de séries temporais no Qlik Sense, consulte *Funções de decomposição de séries temporais (page 1379)*.

Cria um aplicativo

Comece criando um novo aplicativo e importando o conjunto de dados para ele.

Baixe este conjunto de dados:

Tutorial - Decomposição de séries temporais

Esse arquivo contém dados sobre o número de passageiros de uma companhia aérea por mês.

Faça o seguinte:

1. No hub, clique em **Criar novo aplicativo**.
2. Abra o aplicativo e solte o arquivo *Tutorial - Time series decomposition.csv* nele.

Preparar e carregar os dados

Para que o Qlik Sense interprete o campo YearMonth corretamente, talvez seja necessário usar o Gerenciador de dados para reconhecer o campo como um campo de data, e não um campo com valores de string. Em geral, essa etapa é tratada automaticamente. Porém, neste caso, as datas são apresentadas no formato um pouco incomum *YYYY-MM*.

1. No Gerenciador de dados, selecione a tabela e clique em .
2. Com o campo *Ano/Mês* selecionado, clique em e defina o **Tipo de campo** como **Data**.
3. Em **Formato de entrada**, insira *YYYY-MM*.
4. Em **Formato de exibição**, digite *YYYY-MM* e clique em **OK**.
   O campo agora deve mostrar o ícone do calendário.
5. Clique em **Carregar dados**.

Agora, você está pronto para começar a usar as funções STL para representar visualmente seus dados.

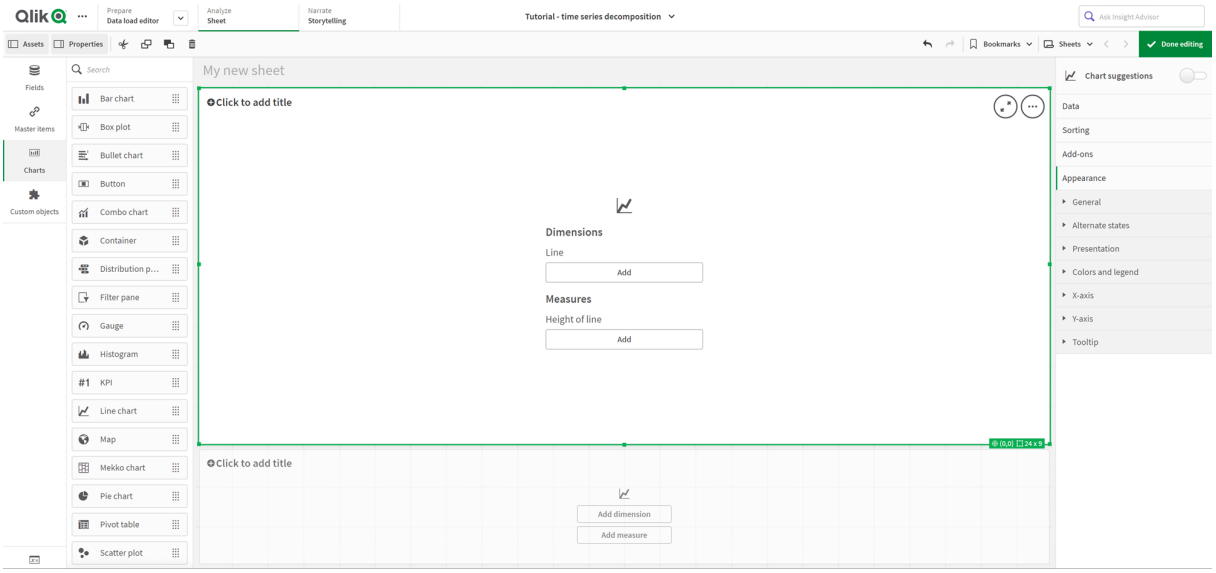
Criar as visualizações

Em seguida, você criará dois gráficos de linhas para demonstrar a funcionalidade das funções de gráfico **STL_Trend**, **STL_Seasonal** e **STL_Residual**.

Abra uma nova pasta e dê um título a ela.

Adicione dois gráficos de linhas à pasta. Redimensione e reposicione os gráficos para que correspondam à imagem a seguir.

Qlik SenseEsboço de grade da pasta de aplicativo em branco



Primeiro gráfico de linhas: Tendência e componentes sazonais

Faça o seguinte:

1. Adicione o título *Sazonal e tendência* ao primeiro gráfico de linhas.
2. Adicione *YearMonth* como uma dimensão e chame-a de *Data*.
3. Adicione a seguinte medida e chame-a de *Passageiros por mês*:
   =Sum(Passengers)
4. Em **Dados**, expanda a medida *Passageiros por mês* e clique em **Adicionar linha de tendências**.
5. Defina **Tipo** como **Linear**.
   Você comparará essa linha de tendências com a saída suavizada do componente de tendência.
6. Adicione a seguinte medida para traçar o componente de tendência e rotule-o como *Tendência*:
   =STL_Trend(SUM(Passengers), 12)
7. Em seguida, adicione a seguinte medida para representar graficamente o componente sazonal e rotule-o como *Sazonal*:
   =STL_Seasonal(SUM(Passengers), 12)
8. Em **Aparência** > **Apresentação**, defina **Barra de rolagem** como **Nenhuma**.
9. Mantenha as cores padrão ou altere-as de acordo com suas preferências.

Segundo gráfico de linhas: Componente residual

Em seguida, configure o segundo gráfico de linhas. Essa visualização exibirá o componente residual da série temporal.

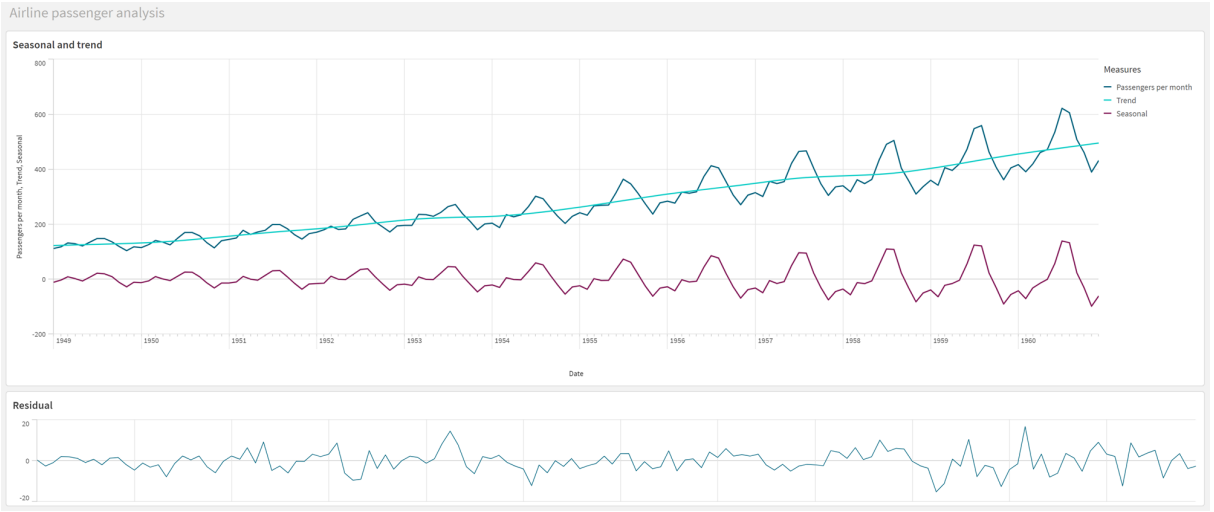
Faça o seguinte:

1. Arraste um gráfico de linhas até a pasta. Adicione o título *Residual*.
2. Adicione *Data* como uma dimensão.

3. Adicione a seguinte medida e chame-a de *Residual*:
   =STL_Residual(SUM(Passengers), 12)
4. Em **Aparência** > **Apresentação**, defina **Barra de rolagem** como **Nenhuma**.

Sua pasta agora deve ter a aparência abaixo.

Pasta Qlik Sense para análise de passageiros de companhias aéreas



Interpretando e explicando os dados

Com as funções do gráfico STL, podemos obter vários insights a partir de nossos dados de séries temporais.

Componente de tendência

As informações estatísticas no componente de tendência são dessazonalizadas. Isso facilita a visualização de flutuações gerais e não repetidas ao longo do tempo. Em comparação com a linha de tendência reta e linear para *Passageiros por mês*, o componente de tendência STL captura as tendências em transformação. Ele exibe alguns desvios claros enquanto ainda apresenta as informações de maneira legível. Os comportamentos de suavização no algoritmo STL ajudaram a capturar isso.

As quedas no número de passageiros de companhias aéreas que são visíveis no gráfico de tendências de STL podem ser explicadas como parte do impacto econômico das recessões que ocorreram durante a década de 1950.

Componente sazonal

O componente sazonal sem tendência isolou flutuações recorrentes ao longo da série temporal e removeu as informações gerais de tendências dessa parte da análise. Começamos com um conjunto de dados que consiste em agregações de ano/mês. Com esses dados, está implícito que estamos segmentando os dados com granularidade de um mês. Ao definir um valor de período de 12, definimos o gráfico para modelar padrões sazonais ao longo de ciclos de um ano (doze meses).

Nos dados, há um padrão sazonal repetido de aumentos de passageiros aéreos nos meses de verão, seguido por quedas nos meses de inverno. Isso está alinhado com a ideia de que o verão é normalmente uma época popular para tirar férias e viajar. Também vemos que, ao longo da série temporal, esses ciclos sazonais aumentam drasticamente em amplitude.

Componente residual

O gráfico do componente residual mostra todas as informações que não foram capturadas na tendência e na decomposição sazonal. O componente residual inclui ruído estatístico, mas também pode indicar uma configuração incorreta da tendência STL e dos argumentos da função sazonal. Geralmente, se houver oscilações periódicas no componente residual do sinal, ou se as informações exibidas claramente não forem aleatórias, geralmente é um sinal de que há informações na série temporal que não estão capturadas atualmente nos componentes sazonais ou de tendência. Nesse caso, você precisa revisar suas definições de cada argumento de função e possivelmente alterar a periodicidade.

Valores mais suaves

Como não especificamos um valor para a tendência e os suavizadores sazonais, a função usará os valores padrão desses parâmetros. Em Qlik Sense, os valores mais suaves padrão no algoritmo STL produzem resultados efetivos. Como resultado, na maioria dos casos, esses argumentos podem ser deixados de fora das expressões.

Definir os argumentos sazonais ou mais suaves de tendência como 0 em qualquer uma das três funções STL faz com que o algoritmo use valores padrão, em vez de valores de 0.

O valor mais suave da tendência usa a dimensão especificada no gráfico. Como o campo *YearMonth* apresenta dados por meses, o valor mais suave da tendência será o número de meses. O suavizador sazonal refletirá a periodicidade definida. Nesse caso, como definimos um período como durando doze meses (um ano), o valor sazonal mais suave é o número de anos. Isso pode parecer confuso, mas na verdade significa que, para encontrarmos a sazonalidade, precisamos examinar várias temporadas. Esse número é o mais suave sazonal.

Outras informações úteis

Como os ciclos sazonais aumentam em amplitude com o tempo, uma abordagem analítica mais avançada poderia fazer uso de funções logarítmicas para criar uma decomposição multiplicativa. Na prática, uma medida simples de amplitude relativa pode ser criada no Qlik Sense dividindo o componente sazonal pelo componente de tendência. Quando isso é feito, notamos que, com o tempo, os picos de verão de cada ciclo aumentam em amplitude relativa. A amplitude dos pontos baixos do inverno, no entanto, não aumenta com o tempo.

5.23 Funções estatísticas de distribuição

Funções de distribuição estatística retornam as probabilidades de ocorrência de diferentes resultados possíveis para uma determinada variável de entrada. Você pode usar essas funções para calcular os valores potenciais dos seus pontos de dados.

Os três grupos de funções de distribuição estatística descritos abaixo são todos implementados no Qlik Sense usando a biblioteca de funções Cephes. Para obter referências e mais detalhes sobre algoritmos usados, precisão, etc., consulte: Cephes library. A biblioteca de funções Cephes é usada com permissão.

- As funções de probabilidade calculam a probabilidade no ponto da distribuição especificado pelo valor fornecido.

    - As funções de Frequência são usadas para distribuições discretas.
    - As funções de Densidade são usadas para funções contínuas.
- As funções Dist calculam a probabilidade acumulada da distribuição no ponto da distribuição especificado pelo valor fornecido.
- As funções Inv calculam o valor inverso, dada a probabilidade acumulada da distribuição.

Todas as funções podem ser usadas no script de carregamento de dados e em expressões de gráfico.

Visão geral das funções de distribuição estatística

Cada função é descrita adicionalmente após a visão geral. Você também pode clicar no nome da função na sintaxe para acessar imediatamente os detalhes dessa função específica.

BetaDensity
`BetaDensity()` retorna a probabilidade da distribuição Beta.

```
BetaDensity (value, alpha, beta)
```

BetaDist
`BetaDist()` retorna a probabilidade acumulada da distribuição Beta.

```
BetaDist (value, alpha, beta)
```

BetaInv
`BetaINV()` retorna o inverso da probabilidade acumulada da distribuição Beta.

```
BetaInv (prob, alpha, beta)
```

BinomDist
`BinomDist()` retorna a probabilidade acumulada da distribuição Binomial.

```
BinomDist (value, trials, trial_probability)
```

BinomFrequency
`BinomFrequency()` retorna a distribuição de probabilidade Binomial.

```
BinomFrequency (value, trials, trial_probability)
```

BinomInv
`BinomInv()` retorna o inverso da probabilidade acumulada da distribuição Binomial.

```
BinomInv (prob, trials, trial_probability)
```

ChiDensity
`ChiDensity()` retorna a probabilidade unicaudal da densidade de distribuição chi^2. A função de densidade chi^2 está associada a um teste de chi^2.

```
ChiDensity (value, degrees_freedom)
```

ChiDist

`ChiDist()` retorna a probabilidade unicaudal da densidade de distribuição chi^2. A distribuição chi^2 está associada a um teste de chi^2.

```
ChiDist (value, degrees_freedom)
```

ChiInv

`ChiInv()` retorna o inverso da probabilidade unicaudal da distribuição de chi^2.

```
ChiInv (prob, degrees_freedom)
```

FDensity

`FDensity()` retorna a probabilidade da distribuição F.

```
FDensity (value, degrees_freedom1, degrees_freedom2)
```

FDist

`FDist()` retorna a probabilidade acumulada da distribuição F.

```
FDist (value, degrees_freedom1, degrees_freedom2)
```

FInv

`FInv()` retorna o inverso da probabilidade acumulada da distribuição de F.

```
FInv (prob, degrees_freedom1, degrees_freedom2)
```

GammaDensity

`GammaDensity()` retorna a probabilidade da distribuição Gamma.

```
GammaDensity (value, k, θ)
```

GammaDist

`GammaDist()` retorna a probabilidade acumulada da distribuição Gamma.

```
GammaDist (value, k, θ)
```

GammaInv

`GammaInv()` retorna o inverso da probabilidade acumulada da distribuição Gamma.

```
GammaInv (prob, k, θ)
```

NormDist

`NormDist()` retorna a distribuição normal acumulada para a média e o desvio padrão especificados. Se mean = 0 e standard_dev = 1, a função retorna a distribuição normal padrão.

```
NormDist (value, mean, standard_dev)
```

NormInv

`NormInv()` retorna o inverso da distribuição normal acumulada para a média e o desvio padrão especificados.

```
NormInv (prob, mean, standard_dev)
```

PoissonDist

`PoissonDist()` retorna a probabilidade acumulada da distribuição de Poisson.

```
PoissonDist (value, mean)
```

PoissonFrequency

`PoissonFrequency()` retorna a distribuição de probabilidade de Poisson.

```
PoissonFrequency (value, mean)
```

PoissonInv

`PoissonInv()` retorna o inverso da probabilidade acumulada da distribuição de Poisson.

```
PoissonInv (prob, mean)
```

TDensity

`TDensity()` retorna o valor da função de densidade de `t` de Student, em que um valor numérico é um valor calculado de `t` cuja probabilidade deve ser calculada.

```
TDensity (value, degrees_freedom, tails)
```

TDist

`TDist()` retorna a probabilidade para a distribuição `t` de Student, em que um valor numérico é um valor calculado de `t` cuja probabilidade deve ser calculada.

```
TDist (value, degrees_freedom, tails)
```

TInv

`TInv()` retorna o valor `t` da distribuição `t` de Student como uma função da probabilidade e os graus de liberdade.

```
TInv (prob, degrees_freedom)
```

Consulte também:

Funções de agregação estatística (page 397)

BetaDensity

`BetaDensity()` retorna a probabilidade da distribuição Beta.

Sintaxe:

```
BetaDensity(value, alpha, beta)
```

Tipo de dados de retorno: número

Argumentos

Argumento	Descrição
`value`	O valor no qual você deseja avaliar a distribuição. O valor deve estar entre 0 e 1.

Argumento	Descrição
alpha	Um número positivo que define o primeiro parâmetro de forma. É o expoente da variável aleatória
beta	Um número positivo que define o segundo parâmetro de forma. Ele indica o número de graus de liberdade do denominador.

BetaDist

`BetaDist()` retorna a probabilidade acumulada da distribuição Beta.

Sintaxe:

```
BetaDist(value, alpha, beta)
```

Tipo de dados de retorno: número

Argumentos

Argumento	Descrição
value	O valor no qual você deseja avaliar a distribuição. O valor deve estar entre 0 e 1.
alpha	Um número positivo que define o primeiro parâmetro de forma. É o expoente da variável aleatória
beta	Um número positivo que define o segundo parâmetro de forma. É o expoente que controla a forma da distribuição.

Esta função está relacionada à função `BetaInv` da seguinte forma:
`If prob = BetaDist(value, alpha, beta), then BetaInv(prob, alpha, beta) = value`

BetaInv

`BetaINV()` retorna o inverso da probabilidade acumulada da distribuição Beta.

Sintaxe:

```
BetaInv(prob, alpha, beta)
```

Tipo de dados de retorno: número

Argumentos

Argumento	Descrição
prob	Uma probabilidade associada à distribuição da probabilidade Beta. Deve ser um número entre 0 e 1.
alpha	Um número positivo que define o primeiro parâmetro de forma. É o expoente da variável aleatória
beta	Um número positivo que define o segundo parâmetro de forma. É o expoente que controla a forma da distribuição.

Esta função está relacionada à função `BetaDist` da seguinte forma:
`If prob = BetaDist(value, alpha, beta), then BetaInv(prob, alpha, beta) = value`

BinomDist

`BinomDist()` retorna a probabilidade acumulada da distribuição Binomial.

Sintaxe:

```
BinomDist(value, trials, trial_probability)
```

Tipo de dados de retorno: número

Argumentos

Argumento	Descrição
`value`	O valor no qual você deseja avaliar a distribuição. O valor deve ser um número inteiro não menor que zero e não maior que o número de tentativas.
`trials`	Um número inteiro positivo que indica o número de tentativas.
`trial_probability`	A probabilidade de sucesso de cada tentativa. É sempre um número entre 0 e 1.

Esta função está relacionada à função `BinomInv` da seguinte forma:
`If prob = BinomDIST(value, trials, trial_probability), then BinomInv(prob, trials, trial_probability) = value`

BinomFrequency

`BinomFrequency()` retorna a distribuição de probabilidade Binomial.

Sintaxe:

```
BinomFrequency(value, trials, trial_probability)
```

Tipo de dados de retorno: número

Argumentos

Argumento	Descrição
`value`	O valor no qual você deseja avaliar a distribuição. O valor deve ser um número inteiro não menor que zero e não maior que o número de tentativas.
`trials`	Um número inteiro positivo que indica o número de tentativas
`trial_probability`	A probabilidade de sucesso de cada tentativa. É sempre um número entre 0 e 1.

BinomInv

`BinomInv()` retorna o inverso da probabilidade acumulada da distribuição Binomial.

Sintaxe:

```
BinomInv(prob, trials, trial_probability)
```

Tipo de dados de retorno: número

Argumentos

Argumento	Descrição
prob	Uma probabilidade associada à distribuição da probabilidade Binomial. Deve ser um número entre 0 e 1.
trials	Um número inteiro positivo que indica o número de tentativas.
trial_probability	A probabilidade de sucesso de cada tentativa. É sempre um número entre 0 e 1.

Esta função está relacionada à função BinomDist da seguinte forma:
`If prob = BinomDist(value, trials, trial_probability), then BinomInv(prob, trials, trial_probability) = value`

ChiDensity

ChiDensity() retorna a probabilidade unicaudal da densidade de distribuição chi^2. A função de densidade chi^2 está associada a um teste de chi^2.

Sintaxe:

```
ChiDensity(value, degrees_freedom)
```

Tipo de dados de retorno: número

Argumentos

Argumento	Descrição
value	O valor no qual você deseja avaliar a distribuição. O valor não deve ser negativo.
degrees_freedom	Um positivo inteiro que declara o número de graus de liberdade do numerador.

ChiDist

ChiDist() retorna a probabilidade unicaudal da densidade de distribuição chi^2. A distribuição chi^2 está associada a um teste de chi^2.

Sintaxe:

```
CHIDIST(value, degrees_freedom)
```

Tipo de dados de retorno: número

Argumentos:

Argumentos

Argumento	Descrição
value	O valor no qual você deseja avaliar a distribuição. O valor não deve ser negativo.
degrees_freedom	Um positivo inteiro que declara o número de graus de liberdade.

Esta função está relacionada à função **ChiInv** da seguinte forma:
`If prob = CHIDIST(value,df), then CHIINV(prob, df) = value`

Limitações:

Todos os argumentos devem ser numéricos, NULL também será retornado.

Exemplos e resultados:

Exemplo	Resultado
CHIDIST(8, 15)	Retorna 0,9238

ChiInv

`ChiInv()` retorna o inverso da probabilidade unicaudal da distribuição de chi^2.

Sintaxe:

```
CHIINV(prob, degrees_freedom)
```

Tipo de dados de retorno: número

Argumentos:

Argumentos

Argumento	Descrição
prob	Uma probabilidade associada à distribuição de chi^2. Deve ser um número entre 0 e 1.
degrees_freedom	Um inteiro que declara o número de graus de liberdade.

Esta função está relacionada à função **ChiDist** da seguinte forma:
`If prob = CHIDIST(value,df), then CHIINV(prob, df) = value`

Limitações:

Todos os argumentos devem ser numéricos, NULL também será retornado.

Exemplos e resultados:

Exemplo	Resultado
CHIINV(0.9237827, 15)	Retorna 8,0000

FDensity

`FDensity()` retorna a probabilidade da distribuição `F`.

Sintaxe:

```
FDensity(value, degrees_freedom1, degrees_freedom2)
```

Tipo de dados de retorno: número

Argumentos

Argumento	Descrição
`value`	O valor no qual você deseja avaliar a distribuição. O valor não deve ser negativo.
`degrees_freedom1`	Um positivo inteiro que declara o número de graus de liberdade do numerador.
`degrees_freedom2`	Um positivo inteiro que declara o número de graus de liberdade do denominador.

FDist

`FDist()` retorna a probabilidade acumulada da distribuição `F`.

Sintaxe:

```
FDist(value, degrees_freedom1, degrees_freedom2)
```

Tipo de dados de retorno: número

Argumentos:

Argumentos

Argumento	Descrição
value	O valor no qual você deseja avaliar a distribuição. O valor não deve ser negativo.
degrees_freedom1	Um positivo inteiro que declara o número de graus de liberdade do numerador.
degrees_freedom2	Um positivo inteiro que declara o número de graus de liberdade do denominador.

Esta função está relacionada à função **FInv** da seguinte forma:
`If prob = FDIST(value, df1, df2), then FINV(prob, df1, df2) = value`

Limitações:

Todos os argumentos devem ser numéricos, NULL também será retornado.

Exemplos e resultados:

Exemplo	Resultado
FDIST(15, 8, 6)	Retorna 0,0019

FInv

`FInv()` retorna o inverso da probabilidade acumulada da distribuição de `F`.

Sintaxe:

```
FInv(prob, degrees_freedom1, degrees_freedom2)
```

Tipo de dados de retorno: número

Argumentos:

Argumentos

Argumento	Descrição
prob	Uma probabilidade associada à distribuição F e deve ser um número entre 0 e 1.
degrees_freedom	Um inteiro que declara o número de graus de liberdade.

Esta função está relacionada à função **FDist** da seguinte forma:
`If prob = FDIST(value, df1, df2), then FINV(prob, df1, df2) = value`

Limitações:

Todos os argumentos devem ser numéricos, NULL também será retornado.

Exemplos e resultados:

Exemplo	Resultado
FINV(0.0019369, 8, 6)	Retorna 15,0000

GammaDensity

`GammaDensity()` retorna a probabilidade da distribuição Gamma.

Sintaxe:

```
GammaDensity(value, k, θ)
```

Tipo de dados de retorno: número

Argumentos

Argumento	Descrição
value	O valor no qual você deseja avaliar a distribuição. O valor não deve ser negativo.
k	Um número positivo que define o parâmetro de forma.
θ	Um número positivo que define o parâmetro de escala.

GammaDist

`GammaDist()` retorna a probabilidade acumulada da distribuição Gamma.

Sintaxe:

```
GammaDist(value, k, θ)
```

Tipo de dados de retorno: número

Argumentos

Argumento	Descrição
value	O valor no qual você deseja avaliar a distribuição. O valor não deve ser negativo.
k	Um número positivo que define o parâmetro de forma.
θ	Um número positivo que define o parâmetro de escala.

Esta função está relacionada à função `GammaINV` da seguinte forma:
`If prob = GammaDist(value, k, θ), then GammaInv(prob, k, θ) = value`

GammaInv

`GammaInv()` retorna o inverso da probabilidade acumulada da distribuição Gamma.

Sintaxe:

```
GammaInv(prob, k, θ)
```

Tipo de dados de retorno: número

Argumentos

Argumento	Descrição
prob	Uma probabilidade associada à distribuição da probabilidade Gama. Deve ser um número entre 0 e 1.
k	Um número positivo que define o parâmetro de forma.
θ	Um número positivo que define o parâmetro de escala.

Esta função está relacionada à função `GammaDist` da seguinte forma:
`If prob = GammaDist(value, k, θ), then GammaInv(prob, k, θ) = value`

NormDist

`NormDist()` retorna a distribuição normal acumulada para a média e o desvio padrão especificados. Se mean = 0 e standard_dev = 1, a função retorna a distribuição normal padrão.

Sintaxe:

```
NORMDIST(value, [mean], [standard_dev], [cumulative])
```

Tipo de dados de retorno: número

Argumentos:

Argumentos

Argumento	Descrição
value	O valor no qual você deseja avaliar a distribuição.
mean	Valor opcional que declara a média aritmética da distribuição. Se você não declarar esse argumento, o valor padrão será 0.
standard_dev	Valor positivo opcional que declara o desvio padrão da distribuição. Se você não declarar esse argumento, o valor padrão será 1.
cumulative	Opcionalmente, você pode optar por usar uma distribuição normal padrão ou uma distribuição cumulativa. 0 = distribuição normal padrão 1 = distribuição cumulativa (padrão)

Esta função está relacionada à função **NormInv** da seguinte forma:
`If prob = NORMDIST(value, m, sd), then NORMINV(prob, m, sd) = value`

Limitações:

Todos os argumentos devem ser numéricos, NULL também será retornado.

Exemplos e resultados:

Exemplo	Resultado
NORMDIST(0.5, 0, 1)	Retorna 0,6915

NormInv

`NormInv()` retorna o inverso da distribuição normal acumulada para a média e o desvio padrão especificados.

Sintaxe:

```
NORMINV(prob, mean, standard_dev)
```

Tipo de dados de retorno: número

Argumentos:

Argumentos

Argumento	Descrição
prob	Uma probabilidade associada à distribuição normal. Deve ser um número entre 0 e 1.
mean	Um valor que declara a média aritmética da distribuição.
standard_dev	Um valor positivo que declara o desvio padrão da distribuição.

Esta função está relacionada à função **NormDist** da seguinte forma:
`If prob = NORMDIST(value, m, sd), then NORMINV(prob, m, sd) = value`

Limitações:

Todos os argumentos devem ser numéricos, NULL também será retornado.

Exemplos e resultados:

Exemplo	Resultado
NORMINV(0.6914625, 0, 1)	Retorna 0,5000

PoissonDist

`PoissonDist()` retorna a probabilidade acumulada da distribuição de Poisson.

Sintaxe:

```
PoissonDist(value, mean)
```

Tipo de dados de retorno: número

Argumentos

Argumento	Descrição
value	O valor no qual você deseja avaliar a distribuição. O valor não deve ser negativo.
mean	Um número positivo que define o resultado médio.

Esta função está relacionada à função `PoissonInv` da seguinte forma:
`If prob = PoissonDist(value, mean), then PoissonInv(prob, mean) = value`

PoissonFrequency

`PoissonFrequency()` retorna a distribuição de probabilidade de Poisson.

Sintaxe:

```
PoissonFrequency(value, mean)
```

Tipo de dados de retorno: número

Argumentos

Argumento	Descrição
value	O valor no qual você deseja avaliar a distribuição. O valor não deve ser negativo.
mean	Um número positivo que define o resultado médio.

PoissonInv

`PoissonInv()` retorna o inverso da probabilidade acumulada da distribuição de Poisson.

Sintaxe:

```
PoissonInv(prob, mean)
```

Tipo de dados de retorno: número

Argumentos

Argumento	Descrição
prob	Uma probabilidade associada à distribuição da probabilidade de Poisson. Deve ser um número entre 0 e 1.
mean	Um número positivo que define o resultado médio.

Esta função está relacionada à função `PoissonDIST` da seguinte forma:
`If prob = PoissonDist(value, mean), then PoissonInv(prob, mean) = value`

TDensity

`TDensity()` retorna o valor da função de densidade de `t` de Student, em que um valor numérico é um valor calculado de `t` cuja probabilidade deve ser calculada.

Sintaxe:

```
TDensity(value, degrees_freedom)
```

Tipo de dados de retorno: número

Argumentos

Argumento	Descrição
value	O valor no qual você deseja avaliar a distribuição. O valor não deve ser negativo.
degrees_freedom	Um positivo inteiro que declara o número de graus de liberdade.

TDist

`TDist()` retorna a probabilidade para a distribuição `t` de Student, em que um valor numérico é um valor calculado de `t` cuja probabilidade deve ser calculada.

Sintaxe:

```
TDist(value, degrees_freedom, tails)
```

Tipo de dados de retorno: número

Argumentos:

Argumentos

Argumento	Descrição
value	O valor no qual você deseja avaliar a distribuição. O valor não deve ser negativo.
degrees_freedom	Um positivo inteiro que declara o número de graus de liberdade.
tails	Deve ser 1 (distribuição de uma extremidade) ou 2 (distribuição de duas extremidades).

Esta função está relacionada à função **TInv** da seguinte forma:
`If prob = TDIST(value, df ,2), then TINV(prob, df) = value`

Limitações:

Todos os argumentos devem ser numéricos, NULL também será retornado.

Exemplos e resultados:

Exemplo	Resultado
TDIST(1, 30, 2)	Retorna 0,3253

TInv

`TInv()` retorna o valor `t` da distribuição `t` de Student como uma função da probabilidade e os graus de liberdade.

Sintaxe:

```
TINV(prob, degrees_freedom)
```

Tipo de dados de retorno: número

Argumentos:

Argumentos

Argumento	Descrição
prob	Uma probabilidade de duas extremidades associada à distribuição t. Deve ser um número entre 0 e 1.
degrees_ freedom	Um inteiro que declara o número de graus de liberdade.

Limitações:

Todos os argumentos devem ser numéricos, NULL também será retornado.

Esta função está relacionada à função **TDist** da seguinte forma:
`If prob = TDIST(value, df ,2), then TINV(prob, df) = value`.

Exemplos e resultados:

Exemplo	Resultado
TINV(0.3253086, 30)	Retorna 1,0000

5.24 Funções de string

Esta seção descreve as funções para lidar e manipular strings.

Todas as funções podem ser usadas no script de carregamento de dados e em expressões do gráfico, com exceção de **Evaluate**, que só pode ser usada no script de carregamento de dados.

Visão geral das funções da cadeia de caracteres

Cada função é descrita adicionalmente após a visão geral. Você também pode clicar no nome da função na sintaxe para acessar imediatamente os detalhes dessa função específica.

Capitalize
Capitalize() retorna a string com todas as palavras em letras iniciais maiúsculas.

```
Capitalize (text)
```

Chr
Chr() retorna o caractere Unicode correspondente ao inteiro de entrada.

```
Chr (int)
```

Evaluate

Evaluate() verifica se a string de texto de entrada pode ser avaliada com uma expressão Qlik Sense válida e, se puder, retorna o valor da expressão com uma string. Se a string de entrada não for uma expressão válida, NULL será retornado.

```
Evaluate (expression_text)
```

FindOneOf

FindOneOf() busca uma string para localizar a posição da ocorrência de qualquer caractere a partir de um conjunto de caracteres fornecidos. A posição da primeira ocorrência de qualquer caractere do conjunto de pesquisa é retornada, a não ser que um terceiro argumento (com um valor maior que 1) seja fornecido. Se não for encontrada correspondência, **0** será retornado.

```
FindOneOf (text, char_set[, count])
```

Hash128

Hash128() retorna um hash de 128 bits dos valores combinados de entrada da expressão. O resultado é uma string de 22 caracteres.

```
Hash128 (expr{, expression})
```

Hash160

Hash160() retorna um hash de 160 bits dos valores combinados de entrada da expressão. O resultado é uma string de 27 caracteres.

```
Hash160 (expr{, expression})
```

Hash256

Hash256() retorna um hash de 256 bits dos valores combinados de entrada da expressão. O resultado é uma string de 43 caracteres.

```
Hash256 (expr{, expression})
```

Index

Index() busca uma string para localizar a posição inicial da enésima ocorrência de uma subsequência fornecida. Um terceiro argumento opcional fornece o valor de n, que é 1 se for omitido. Um valor negativo busca a partir do final da string. As posições nos caracteres são numeradas de **1** em diante.

```
Index  (text, substring[, count])
```

IsJson

IsJson() testa se uma string especificada contém dados JSON (JavaScript Object Notation) válidos. Você também pode validar um tipo de dados JSON específico.

```
IsJson (json [, type])
```

JsonGet

JsonGet() retorna o caminho de uma string de dados JSON (JavaScript Object Notation). Os dados devem ser JSON válidos, mas podem conter espaços extras ou novas linhas.

```
JsonGet (json, path)
```

JsonSet

JsonSet() modifica uma string contendo dados JSON (JavaScript Object Notation). Ele pode definir ou inserir um valor JSON com o novo local especificado pelo caminho. Os dados devem ser JSON válidos, mas podem conter espaços extras ou novas linhas.

```
JsonSet(json, path, value)
```

KeepChar

KeepChar() retorna uma string que consiste na primeira string, "text", menos qualquer caractere não contido na segunda string, "keep_chars".

```
KeepChar (text, keep_chars)
```

Left

Left() retorna uma string que consiste nos primeiros caracteres (os mais à esquerda) da string de entrada, em que o número de caracteres é determinado pelo segundo argumento.

```
Left (text, count)
```

Len

Len() retorna o comprimento da cadeia de caracteres de entrada.

```
Len (text)
```

LevenshteinDist

LevenShteInst() retorna a distância Levenshtein entre duas strings. Ela é definida como o número mínimo de edições de caractere único (inserções, exclusões ou substituições) necessárias para transformar uma string na outra. Essa função é útil para comparações de strings difusas.

```
LevenshteinDist (text1, text2)
```

Lower

Lower() converte todos os caracteres na string de entrada em minúsculos.

```
Lower (text)
```

LTrim

LTrim() retorna a string de entrada destituída de espaços à esquerda.

```
LTrim (text)
```

Mid

Mid() retorna a parte da string iniciando na posição do caractere definido pelo segundo argumento, "start", com o comprimento da string definido pelo terceiro argumento, '"count". Se "count" for omitido, será retornado o restante da string de entrada. O primeiro caractere na string de entrada é o de número 1.

```
Mid (text, start[, count])
```

Ord

Ord() retorna o número de ponto do código Unicode do primeiro caractere da string de entrada.

```
Ord (text)
```

PurgeChar

PurgeChar() retorna uma string contendo todos os caracteres na string de entrada ('text'), exceto para qualquer caractere presente no segundo argumento ('remove_chars').

```
PurgeChar (text, remove_chars)
```

Repeat

Repeat() forma uma string que consiste em uma string de entrada, repetido o número de vezes definido pelo segundo argumento.

```
Repeat (text[, repeat_count])
```

Replace

Replace() retorna uma string depois de substituir todas as ocorrências de uma determinada subsequência dentro da string de entrada com outra subsequência. A função não é recursiva e funciona da esquerda para a direita.

```
Replace (text, from_str, to_str)
```

Right

Right() retorna uma string que consiste nos últimos caracteres (mais à direita) da string de entrada, em que o número de caracteres é determinado pelo segundo argumento.

```
Right (text, count)
```

RTrim

RTrim() retorna a string de entrada destituída de espaços à direita.

```
RTrim (text)
```

SubField

SubField() é usado para extrair componentes de substring a partir de um campo primário de string, em que os campos de registro originais consistem em duas ou mais partes separadas por um separador.

```
SubField (text, delimiter[, field_no ])
```

SubStringCount

SubStringCount() retorna o número de ocorrências da subsequência especificada no texto da string de entrada. Se não houver uma correspondência, será retornado 0.

```
SubStringCount (text, substring)
```

TextBetween

TextBetween() retorna o texto na string de entrada que ocorre entre os caracteres especificados como delimitadores.

```
TextBetween (text, delimiter1, delimiter2[, n])
```

Trim

Trim() retorna a string de entrada destituída de espaços à esquerda e à direita.

```
Trim (text)
```

Upper

Upper() converte todos os caracteres na string de saída para maiúsculos em todos os caracteres de texto na expressão. Números e símbolos são ignorados.

```
Upper (text)
```

Capitalize

Capitalize() retorna a string com todas as palavras em letras iniciais maiúsculas.

Sintaxe:

```
Capitalize(text)
```

Tipo de dados de retorno: caractere

Exemplo: Expressões de gráfico

Exemplo	Resultado
`Capitalize ( 'star trek' )`	Retorna 'Star Trek'
`Capitalize ( 'AA bb cC Dd')`	Retorna 'Aa Bb Cc Dd'

Exemplo: Script de carregamento

```
Load  String, Capitalize(String) Inline [String rHode iSland washingTon d.C. new york];
```

Resultado

Cadeia	Capitalize(String)
rHode iSland	Rhode Island
washingTon d.C.	Washington D.C.
new york	New York

Chr

Chr() retorna o caractere Unicode correspondente ao inteiro de entrada.

Sintaxe:

```
Chr(int)
```

Tipo de dados de retorno: caractere

Exemplos e resultados:

Exemplo	Resultado
Chr(65)	Retorna a string 'A'

Exemplo	Resultado
Chr(163)	Retorna a string '£'
Chr(35)	Retorna a string '#'

Evaluate

Evaluate() verifica se a string de texto de entrada pode ser avaliada com uma expressão Qlik Sense válida e, se puder, retorna o valor da expressão com uma string. Se a string de entrada não for uma expressão válida, NULL será retornado.

Sintaxe:

```
Evaluate(expression_text)
```

Tipo de dados de retorno: dual

Essa função de string não pode ser utilizada em expressões de gráfico.

Exemplos e resultados:

Exemplo de função	Resultado
`Evaluate ( 5 * 8 )`	Retorna '40'

Exemplo de script de carregamento

```
Load  Evaluate(String) as Evaluated, String Inline [String 4 5+3 0123456789012345678 Today()
];
```

Resultado

Cadeia	Avaliado
4	4
5+3	8
0123456789012345678	0123456789012345678
Today()	2022-02-02

FindOneOf

FindOneOf() busca uma string para localizar a posição da ocorrência de qualquer caractere a partir de um conjunto de caracteres fornecidos. A posição da primeira ocorrência de qualquer caractere do conjunto de pesquisa é retornada, a não ser que um terceiro argumento (com um valor maior que 1) seja fornecido. Se não for encontrada correspondência, **0** será retornado.

Sintaxe:

```
FindOneOf(text, char_set[, count])
```

Tipo de dados de retorno: inteiro

Argumentos:

Argumentos

Argumento	Descrição
text	A string original.
char_set	Um conjunto de caracteres para pesquisar em text.
count	Define qual ocorrência de qualquer caractere para pesquisar. Por exemplo, um valor de duas pesquisas para a segunda ocorrência.

Exemplo: Expressões de gráfico

Exemplo	Resultado
`FindOneOf( 'my example text string', 'et%s')`	Retorna '4' porque 'e' é o quarto caractere na string de exemplo.
`FindOneOf( 'my example text string', 'et%s', 3)`	Retorna '12' porque a pesquisa refere-se a qualquer um dos caracteres e, t, % ou s, e "t" é a terceira ocorrência na posição 12 da string de exemplo.
`FindOneOf( 'my example text string', '¤%&')`	Retorna '0' porque nenhum dos caracteres ¤, % ou & existe na string de exemplo.

Exemplo: Script de carregamento

```
Load * Inline [SearchFor, Occurrence et%s,1 et%s,3 ¤%&,1]
```

Resultado

SearchFor	Ocorrência	FindOneOf('my example text string', SearchFor, Occurrence)
et%s	1	4
et%s	3	12
¤%&	1	0

Hash128

Hash128() retorna um hash de 128 bits dos valores combinados de entrada da expressão. O resultado é uma string de 22 caracteres.

Sintaxe:

```
Hash128(expr{, expression})
```

Tipo de dados de retorno: caractere

Exemplo: Expressões de gráfico

Exemplo	Resultado
Hash128 ('abc', 'xyz', '123')	Retorna 'MA&5]6+3=:>:>G%S<U*S2+'.
Hash128 (Region, Year, Month) Note: Region, Year, and Month are table fields.	Retorna 'G7*=6GKPJ(Z+)^KM?<$'A+'.

Exemplo: Script de carregamento

```
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 ];
```

Resultado

Região	Ano	Mês	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() retorna um hash de 160 bits dos valores combinados de entrada da expressão. O resultado é uma string de 27 caracteres.

Sintaxe:

```
Hash160(expr{, expression})
```

Tipo de dados de retorno: caractere

Exemplo: Expressões de gráfico

Exemplo	Resultado
Hash160 ('abc', 'xyz', '123')	Retorna 'MA&5]6+3=:>;>G%S<U*S2I:`=X*'.
Hash160 (Region, Year, Month) Note: Region, Year, and Month are table fields.	Retorna 'G7*=6GKPJ (Z+)^KM?<$'AI.)?U$'.

Exemplo: Script de carregamento

```
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 ];
```

Resultado

Região	Ano	Mês	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() retorna um hash de 256 bits dos valores combinados de entrada da expressão. O resultado é uma string de 43 caracteres.

Sintaxe:

```
Hash256(expr{, expression})
```

Tipo de dados de retorno: caractere

Exemplo: Expressões de gráfico

Exemplo	Resultado
Hash256 ('abc', 'xyz', '123')	Retorna 'MA&5]6+3=:>;>G%S<U*S2I:`=X*A.IO*8N\%Y7Q;YEJ'.
Hash256 (Region, Year, Month) Note: Region, Year, and Month are table fields.	Retorna 'G7*=6GKPJ(Z+)^KM?<$'AI.)?U$#X2RB [:0ZP=+Z`F:'.

Exemplo: Script de carregamento

```
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 ];
```

Resultado

Região	Ano	Mês	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() busca uma string para localizar a posição inicial da enésima ocorrência de uma subsequência fornecida. Um terceiro argumento opcional fornece o valor de n, que é 1 se for omitido. Um valor negativo busca a partir do final da string. As posições nos caracteres são numeradas de **1** em diante.

Sintaxe:

```
Index(text, substring[, count])
```

Tipo de dados de retorno: inteiro

Argumentos:

Argumentos

Argumento	Descrição
text	A string original.
substring	Uma string de caracteres para pesquisar em text.
count	Define qual ocorrência de **substring** pesquisar. Por exemplo, um valor de duas pesquisas para a segunda ocorrência.

Exemplos e resultados:

Exemplo	Resultado
Index('abcdefg', 'cd')	Retorna 3
Index('abcdabcd', 'b', 2)	Retorna 6 (a segunda ocorrência de "b")
Index('abcdabcd', 'b',-2)	Retorna 2 (a segunda ocorrência de "b" começando do final)
`Left( Date, Index( Date,'-') -1 )` where **Date** = 1997-07-14	Retorna 1997
`Mid( Date, Index( Date, '-', 2 ) -2, 2 )` where **Date** = 1997-07-14	Retorna 07

Exemplo: Script

```
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() testa se uma string especificada contém dados JSON (JavaScript Object Notation) válidos. Você também pode validar um tipo de dados JSON específico.

Sintaxe:

```
value IsJson(json [, type])
```

Tipo de dados de retorno: dual

Argumentos

Argumento	Descrição
json	String a ser testada. Pode conter espaços extras ou novas linhas.
type	Argumento opcional que especifica o tipo de dados JSON a ser testado. • 'value' (padrão) • 'object' • 'array' • 'string' • 'number' • 'Boolean' • 'null'

Exemplo: JSON e tipo válidos

Exemplo	Resultado
`IsJson('null')`	Retorna -1 (true)
`IsJson('"abc"', 'value')`	Retorna -1 (true)
`IsJson('"abc"', 'string')`	Retorna -1 (true)
`IsJson(123, 'number')`	Retorna -1 (true)

Exemplo: JSON ou tipo inválido

Exemplo	Resultado	Descrição
`IsJson('text')`	Retorna 0 (false)	'text' não é um valor JSON válido
`IsJson('"text"', 'number')`	Retorna 0 (false)	'"text"' não é um número JSON válido
`IsJson('"text"', 'text')`	Retorna 0 (false)	'text' não é um tipo JSON válido

JsonGet

JsonGet() retorna o caminho de uma string de dados JSON (JavaScript Object Notation). Os dados devem ser JSON válidos, mas podem conter espaços extras ou novas linhas.

Sintaxe:

```
value JsonGet(json, path)
```

Tipo de dados de retorno: dual

Argumentos

Argumento	Descrição
json	String contendo dados JSON.
path	O caminho deve ser especificado de acordo com a RFC 6901. Isso permitirá a pesquisa de propriedades dentro de dados JSON sem usar funções complexas de substring ou índice.

Exemplo: Caminho e JSON válidos

Exemplo	Resultado
`JsonGet('{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}', '')`	Retorna '{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}'
`JsonGet('{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}', '/a')`	Retorna '{"foo":"bar"}'
`JsonGet('{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}', '/a/foo')`	Retorna '"bar"'
`JsonGet('{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}', '/b')`	Retorna '[123,"abc","ABC"]'
`JsonGet('{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}', '/b/0')`	Retorna '123'
`JsonGet('{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}', '/b/1')`	Retorna '"abc"'
`JsonGet('{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}', '/b/2')`	Retorna '"ABC"'

Exemplo: Caminho ou JSON inválido

Exemplo	Resultado	Descrição
`JsonGet('{"a":"b"}','/b')`	Retorna null	O caminho não aponta para uma parte válida dos dados JSON.
`JsonGet('{"a"}','/a')`	Retorna null	Os dados JSON não são válidos (o membro "a" não tem um valor).

JsonSet

JsonSet() modifica uma string contendo dados JSON (JavaScript Object Notation). Ele pode definir ou inserir um valor JSON com o novo local especificado pelo caminho. Os dados devem ser JSON válidos, mas podem conter espaços extras ou novas linhas.

Sintaxe:

```
value JsonSet(json, path, value)
```

Tipo de dados de retorno: dual

Argumentos

Argumento	Descrição
json	String contendo dados JSON.
path	O caminho deve ser especificado de acordo com a RFC 6901. Isso permite o acúmulo de propriedades dentro de dados JSON sem usar funções complexas de substring ou índice e concatenação.
value	O novo valor da string em formato JSON.

Exemplo: JSON, caminho e valor válidos

Exemplo	Resultado
`JsonSet('{}','/a','"b"')`	Retorna '{"a":"b"}'
`JsonSet('[]','/0','"x"')`	Retorna '["x"]'
`JsonSet('"abc"','','123')`	Retorna 123

Exemplo: JSON, caminho ou valor inválido

Exemplo	Resultado	Descrição
`JsonSet('"abc"','/x','123')`	Retorna null	O caminho não aponta para uma parte válida dos dados JSON.
`JsonSet('{"a": {"b":"c"}}','a/b','"x"')`	Retorna null	O caminho é inválido.
`JsonSet(' {"a":"b"}','/a','abc')`	Retorna null	O valor não é um JSON válido. Uma string deve estar entre aspas.

KeepChar

KeepChar() retorna uma string que consiste na primeira string, "text", menos qualquer caractere não contido na segunda string, "keep_chars".

Sintaxe:

```
KeepChar(text, keep_chars)
```

Tipo de dados de retorno: caractere

Argumentos:

Argumentos

Argumento	Descrição
text	A string original.
keep_chars	Uma string contendo os caracteres no text a ser mantido.

Exemplo: Expressões de gráfico

Exemplo	Resultado
`KeepChar ( 'a1b2c3','123' )`	Retorna '123'.
`KeepChar ( 'a1b2c3','1234' )`	Retorna '123'.
`KeepChar ( 'a1b22c3','1234' )`	Retorna '1223'.
`KeepChar ( 'a1b2c3','312' )`	Retorna '123'.

Exemplo: Script de carregamento

```
T1:
Load
*,
keepchar(String1, String2) as KeepChar;
Load * inline [
String1, String2
'a1b2c3', '123'
];
```

Resultados

Tabela do Qlik Sense mostrando a saída do uso da função *KeepChar* no script de carregamento.

String1	String2	KeepChar
a1b2c3	123	123

Consulte também:

PurgeChar (page 1471)

Left

Left() retorna uma string que consiste nos primeiros caracteres (os mais à esquerda) da string de entrada, em que o número de caracteres é determinado pelo segundo argumento.

Sintaxe:

```
Left(text, count)
```

Tipo de dados de retorno: caractere

Argumentos:

Argumento	Descrição
text	A string original.
count	Define o número de caracteres a ser incluído a partir da parte esquerda da string **text**.

Exemplo: Expressão de gráfico

Exemplo	Resultado
`Left('abcdef', 3)`	Retorna 'abc'

Exemplo: Script de carregamento

```
T1: Load  *,  left(Text,Start) as Left;        Load * inline [ Text, Start 'abcdef', 3 '2021-07-14', 4 '2021-07-14', 2 ];
```

Resultado

Tabela do Qlik Sense mostrando a saída do uso da função *Left* no script de carregamento.

Texto	Início	Left
abcdef	3	abc
2021-07-14	4	2021
2021-07-14	2	20

Consulte também *Index (page 1461)*, que permite uma análise de string mais complexa.

Len

Len() retorna o comprimento da cadeia de caracteres de entrada.

Sintaxe:

```
Len(text)
```

Tipo de dados de retorno: inteiro

Exemplo: Expressão de gráfico

Exemplo	Resultado
`Len('Peter')`	Retorna '5'

Exemplo: Script de carregamento

```
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 ];
```

Resultado

Cadeia	NewString
this is a sample text string	This is a sample text string
capitalize first letter only	Capitalize first letter only

LevenshteinDist

LevenShteInst() retorna a distância Levenshtein entre duas strings. Ela é definida como o número mínimo de edições de caractere único (inserções, exclusões ou substituições) necessárias para transformar uma string na outra. Essa função é útil para comparações de strings difusas.

Sintaxe:

```
LevenshteinDist(text1, text2)
```

Tipo de dados de retorno: inteiro

Exemplo: Expressão de gráfico

Exemplo	Resultado
`LevenshteinDist('Kitten','Sitting')`	Retorna "3"

Exemplo: Script de carregamento

Script de carregamento

```
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;
```

Resultado

ID	String_1	String_2	LevenshteinDistance
1	Silver	Sliver	2
2	Silver	SSiver	2
3	Silver	SSiveer	3
4	Gold	Negrito	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() converte todos os caracteres na string de entrada em minúsculos.

Sintaxe:

```
Lower(text)
```

Tipo de dados de retorno: caractere

Exemplo: Expressão de gráfico

Exemplo	Resultado
`Lower('abcD')`	Retorna 'abcd'

Exemplo: Script de carregamento

```
Load  String, Lower(String) Inline [String rHode iSland washingTon d.C. new york];
```

Resultado

Cadeia	Lower(String)
rHode iSland	rhode island
washingTon d.C.	washington d.c.
new york	new york

LTrim

LTrim() retorna a string de entrada destituída de espaços à esquerda.

Sintaxe:

```
LTrim(text)
```

Tipo de dados de retorno: caractere

Exemplo: Expressões de gráfico

Exemplo	Resultado
`LTrim( ' abc' )`	Retorna 'abc'
`LTrim( 'abc ' )`	Retorna 'abc '

Exemplo: Script de carregamento

```
Set verbatim=1;  T1:  Load *, len(LtrimString) as LtrimStringLength;       Load *, ltrim
(String) as LtrimString; Load *, len(String) as StringLength;                Load * Inline [
String '   abc    ' ' def  '];
```

A instrução "Set verbatim=1" está incluída no exemplo para garantir que os espaços não sejam aparados automaticamente antes da demonstração da função ltrim. Para obter mais informações, consulte Verbatim (page 207).

Resultado

Cadeia	StringLength	LtrimStringLength
def	6	5
abc	10	7

Consulte também:

- *RTrim (page 1475)*

Mid

Mid() retorna a parte da string iniciando na posição do caractere definido pelo segundo argumento, "start", com o comprimento da string definido pelo terceiro argumento, '"count". Se "count" for omitido, será retornado o restante da string de entrada. O primeiro caractere na string de entrada é o de número 1.

Sintaxe:

```
Mid(text, start[, count])
```

Tipo de dados de retorno: caractere

Argumentos:

Argumentos

Argumento	Descrição
text	A string original.
start	Inteiro definindo a posição do primeiro caractere em text a ser incluído.
count	Define o comprimento da string de saída. Se omitidos, todos os caracteres a partir da posição definida pelo **start** são incluídos.

Exemplo: Expressões de gráfico

Exemplo	Resultado
`Mid('abcdef',3 )`	Retorna 'cdef'
`Mid('abcdef',3, 2 )`	Retorna 'cd'

Exemplo: Script de carregamento

```
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 ];
```

Resultado

Tabela do Qlik Sense mostrando a saída do uso da função *Mid* no script de carregamento.

Texto	Início	Mid1	Contagem	Mid2
abcdef	2	bcdef	3	bcd
abcdef	3	cdef	2	cd
210714	2	10714	3	107
210714	3	0714	2	07

Consulte também:

Index (page 1461)

Ord

Ord() retorna o número de ponto do código Unicode do primeiro caractere da string de entrada.

Sintaxe:

```
Ord(text)
```

Tipo de dados de retorno: inteiro

Exemplos e resultados:

Exemplo: Expressão de gráfico

Exemplo	Resultado
`Ord('A')`	Retorna o inteiro 65.
`Ord('Ab')`	Retorna o inteiro 65.

Exemplo: Script de carregamento

```
//Guqin (Chinese: 古琴) – 7-stringed zithers T2: Load *,  ord(Chinese) as OrdUnicode,
        ord(Western) as OrdASCII;                    Load * inline [ Chinese, Western 古琴,
Guqin ];
```

Resultado:

Chinês	Ocidental	OrdASCII	OrdUnicode
古琴	Guqin	71	21476

PurgeChar

PurgeChar() retorna uma string contendo todos os caracteres na string de entrada ('text'), exceto para qualquer caractere presente no segundo argumento ('remove_chars').

Sintaxe:

```
PurgeChar(text, remove_chars)
```

Tipo de dados de retorno: caractere

Argumentos:

Argumentos

Argumento	Descrição
text	A string original.
remove_chars	Uma string contendo os caracteres no text a ser removido.

Tipo de dados de retorno: caractere

Exemplo: Expressões de gráfico

Exemplo	Resultado
`PurgeChar ( 'a1b2c3','123' )`	Retorna 'abc'.
`PurgeChar ( 'a1b2c3','312' )`	Retorna 'abc'.

Exemplo: Script de carregamento

```
T1:
Load
*,
purgechar(String1, String2) as PurgeChar;
Load * inline [
String1, String2
'a1b2c3', '123'
];
```

Resultados

Tabela do Qlik Sense mostrando a saída do uso da função *PurgeChar* no script de carregamento.

String1	String2	PurgeChar
a1b2c3	123	abc

Consulte também:

KeepChar (page 1464)

Repeat

Repeat() forma uma string que consiste em uma string de entrada, repetido o número de vezes definido pelo segundo argumento.

Sintaxe:

```
Repeat(text[, repeat_count])
```

Tipo de dados de retorno: caractere

Argumentos:

Argumentos

Argumento	Descrição
text	A string original.
repeat_count	Define o número de vezes que os caracteres na string **text** serão repetidos na string de saída.

Exemplo: Expressão de gráfico

Exemplo	Resultado
`Repeat( ' * ', rating )` when **rating** = 4	Retorna '****'

Exemplo: Script de carregamento

```
T1: Load *, repeat(String,2) as Repeat; Load * inline [ String hello world! hOw aRe you? ];
```

Resultado

Cadeia	Repetir
hello world!	hello world!hello world!
hOw aRe you?	hOw aRe you?hOw aRe you?

Replace

Replace() retorna uma string depois de substituir todas as ocorrências de uma determinada subsequência dentro da string de entrada com outra subsequência. A função não é recursiva e funciona da esquerda para a direita.

Sintaxe:

```
Replace(text, from_str, to_str)
```

Tipo de dados de retorno: caractere

Argumentos:

Argumentos

Argumento	Descrição
text	A string original.
from_str	Uma string que pode ocorrer uma ou mais vezes dentro da string de entrada **text**.
to_str	A string que substituirá todas as ocorrências de **from_str** dentro da string **text**.

Exemplos e resultados:

Exemplo	Resultado
`Replace('abccde','cc','xyz')`	Retorna 'abxyzde'

Consulte também:

Right

Right() retorna uma string que consiste nos últimos caracteres (mais à direita) da string de entrada, em que o número de caracteres é determinado pelo segundo argumento.

Sintaxe:

```
Right(text, count)
```

Tipo de dados de retorno: caractere

Argumentos:

Argumentos

Argumento	Descrição
text	A string original.
count	Define o número de caracteres a serem incluídos na parte mais à direita da string **text**.

Exemplo: Expressão de gráfico

Exemplo	Resultado
`Right('abcdef', 3)`	Retorna 'def'

Exemplo: Script de carregamento

```
T1:
Load
*,
right(Text,Start) as Right;
Load * inline [
Text, Start
'abcdef', 3
'2021-07-14', 4
'2021-07-14', 2
];
```

Resultado

Tabela do Qlik Sense mostrando a saída do uso da função *Right* no script de carregamento.

Texto	Início	Right
abcdef	3	def
2021-07-14	4	7-14
2021-07-14	2	14

RTrim

RTrim() retorna a string de entrada destituída de espaços à direita.

Sintaxe:

```
RTrim(text)
```

Tipo de dados de retorno: caractere

Exemplo: Expressões de gráfico

Exemplo	Resultado
`RTrim( ' abc' )`	Retorna ' abc'
`RTrim( 'abc ' )`	Retorna 'abc'

Exemplo: Script de carregamento

```
Set verbatim=1;    T1:  Load *, len(RtrimString) as RtrimStringLength;         Load *, rtrim
(String) as RtrimString; Load *, len(String) as StringLength;              Load * Inline [
String '   abc    ' ' def  '];
```

A instrução "Set verbatim=1" está incluída no exemplo para garantir que os espaços não sejam aparados automaticamente antes da demonstração da função rtrim. Para obter mais informações, consulte Verbatim (page 207).

Resultado

Cadeia	StringLength	RtrimStringLength
def	6	4
abc	10	6

Consulte também:

- *LTrim (page 1469)*

SubField

SubField() é usado para extrair componentes de substring a partir de um campo primário de string, em que os campos de registro originais consistem em duas ou mais partes separadas por um separador.

A função **Subfield()** pode ser usada, por exemplo, para extrair o primeiro nome e sobrenome de uma lista de registros que consiste em nomes completos, partes do componente de um nome de caminho ou para extrair dados de tabelas separadas por vírgulas.

Se você usar a função **Subfield()** em um comando de **LOAD** com o parâmetro field_no opcional deixado de fora, um registro completo será gerado para cada substring. Se vários campos forem carregados usando **Subfield()**, os produtos cartesianos de todas as combinações serão criados.

Sintaxe:

```
SubField(text, delimiter[, field_no ])
```

Tipo de dados de retorno: caractere

Argumentos:

Argumentos

Argumento	Descrição
text	A string original. Isso pode ser um texto codificado, uma variável, uma expansão de sinal de dólar ou outra expressão.
delimiter	Um caractere dentro do **text** de entrada que divide a string em partes de componente.
field_no	O terceiro argumento opcional é um inteiro que especifica qual substring do **text** da string primária será retornado. Use o valor 1 para retornar a primeira subcadeia, 2 para retornar a segunda subcadeia, e assim por diante. • Se **field_no** for um valor positivo, as subcadeias serão extraídas da esquerda para a direita. • Se **field_no** for um valor negativo, as subcadeias serão extraídas da direita para a esquerda.

SubField() pode ser usado em vez de combinações complexas de funções, tais como Len(), Right(), Left(), Mid() e outras funções de caracteres.

Exemplos: Expressões de script e gráfico usando SubField

Exemplos - expressões de script e gráfico

Exemplos básicos

Exemplo	Resultado
`SubField(S, ';' ,2)`	Retorna 'cde' se **S** for 'abc;cde;efg'.
`SubField(S, ';' ,1)`	Retorna uma string vazia quando **S** é uma string vazia.
`SubField(S, ';' ,1)`	Retorna uma string vazia se **S** for ';'.
Imagine que você tem uma variável com o nome de caminho vMyPath, `Set vMyPath=\Users\ext_jrb\Documents\Qlik\Sense\Apps;.`	Em um gráfico de texto e imagem, é possível adicionar uma medida como: `SubField(vMyPath, '\',-3)`, que resulta em 'Qlik', porque é a terceira substring a contar da extremidade direita da variável vMyPath.

Exemplo de script 1

Script de carregamento

Carregue as seguintes expressões de script e dados no editor de carregamento de dados.

```
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;
```

Criar uma visualização

Crie uma visualização de tabela em uma Qlik Sense do Qlik Cloud com **Name**, **FirstName** e **SurName** como dimensões

Resultado

Name	FirstName	SurName
Dave Owen	Dave	Owen
Joe Tem	Joe	Tem

Explicação

A função **SubField()** extrai a primeira substring de **Name** definindo o argumento **field_no** como 1. Como o valor de **field_no** é positivo, uma ordem da esquerda para a direita é seguida para extrair a subtring. Uma segunda chamada de função extrai a segunda substring definindo o argumento **field_no** como -1, que extrai a substring seguindo uma ordem da direita para a esquerda.

Exemplo de script 2

Script de carregamento

Carregue as seguintes expressões de script e dados no editor de carregamento de dados.

```
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 '|');
```

Criar uma visualização

Crie uma visualização de tabela em uma pasta do Qlik Sense com **Instrument**, **Player** e **Project** como dimensões.

Resultado

Instrument	Player	Project
Guitar	Mike	Music
Guitar	Mike	Video
Guitar	Mike	OST
Guitar	Neil	Music
Guitar	Neil	Video
Guitar	Neil	OST
Synth	Jen	Music
Synth	Jen	Video
Synth	Jen	OST
Synth	Jo	Music

Instrument	Player	Project
Synth	Neil	Music
Synth	Neil	Video
Synth	Neil	OST

Explicação

Esse exemplo mostra como usar várias instâncias da função **Subfield()**, cada uma com o parâmetro field_no deixado de fora, a partir do mesmo comando **LOAD**, cria produtos Cartesianos de todas as combinações. A opção **DISTINCT** é usada para evitar a criação de registros duplicados.

SubStringCount

SubStringCount() retorna o número de ocorrências da subsequência especificada no texto da string de entrada. Se não houver uma correspondência, será retornado 0.

Sintaxe:

```
SubStringCount(text, sub_string)
```

Tipo de dados de retorno: inteiro

Argumentos:

Argumento	Descrição
text	A string original.
sub_string	Uma string que pode ocorrer uma ou mais vezes dentro da string de entrada **text**.

Exemplo: Expressões de gráfico

Exemplo	Resultado
`SubStringCount ( 'abcdefgcdxyz', 'cd' )`	Retorna '2'
`SubStringCount ( 'abcdefgcdxyz', 'dc' )`	Retorna '0'

Exemplo: Script de carregamento

```
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 ];
```

Resultado

Sequências de caracteres	SubStringCount_AB
aB/cd/ef/gh/Abc/abandoned	3
ABC:DEF:GHI:AB:CD:EF:GH	2

TextBetween

TextBetween() retorna o texto na string de entrada que ocorre entre os caracteres especificados como delimitadores.

Sintaxe:

```
TextBetween(text, delimiter1, delimiter2[, n])
```

Tipo de dados de retorno: caractere

Argumentos:

Argumento	Descrição
text	A string original.
delimiter1	Especifica o primeiro caractere delimitador (ou string) para pesquisar em **text**.
delimiter2	Especifica o segundo caractere delimitador (ou string) para pesquisar em **text**.
n	Define entre qual ocorrência do par separador pesquisar. Por exemplo, um valor de 2 retorna os caracteres entre a segunda ocorrência do delimiter1 e a segunda ocorrência do delimiter2.

Exemplo: Expressões de gráfico

Exemplo	Resultado
`TextBetween('<abc>', '<', '>')`	Retorna 'abc'
`TextBetween('<abc><de>', '<', '>',2)`	Retorna 'de'
`TextBetween('abc', '<', '>')` `TextBetween('<a<b', '<', '>')`	Ambos os exemplos retornam NULL. Se algum delimitador não for encontrado na cadeia, NULL será retornado.
`TextBetween('<>', '<', '>')`	Retorna uma cadeia de comprimento zero.
`TextBetween('<abc>', '<', '>', 2)`	Retorna NULL, pois n é maior que o número de ocorrências dos delimitadores.

Exemplo: Script de carregamento

```
Load *, textbetween(Text,'<','>') as TextBetween, textbetween(Text,'<','>',2) as
SecondTextBetween; Load * inline [ Text <abc><de> <def><ghi><jkl> ];
```

Resultado

Texto	TextBetween	SecondTextBetween
<abc><de>	abc	de
<def><ghi><jkl>	def	ghi

Trim

Trim() retorna a string de entrada destituída de espaços à esquerda e à direita.

Sintaxe:

```
Trim(text)
```

Tipo de dados de retorno: caractere

Exemplos e resultados:

Exemplo: Expressão de gráfico

Exemplo	Resultado
`Trim( ' abc' )`	Retorna 'abc'
`Trim( 'abc ' )`	Retorna 'abc'
`Trim( ' abc ' )`	Retorna 'abc'

Exemplo: Script de carregamento

```
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');
```

A instrução "Set verbatim=1" está incluída no exemplo para garantir que os espaços não sejam aparados automaticamente antes da demonstração da função trim. Para obter mais informações, consulte Verbatim (page 207).

Resultado:

Cadeia	StringLength	TrimStringLength
def	6	3
abc	10	3

Upper

Upper() converte todos os caracteres na string de saída para maiúsculos em todos os caracteres de texto na expressão. Números e símbolos são ignorados.

Sintaxe:

```
Upper(text)
```

Tipo de dados de retorno: caractere

Exemplo: Expressão de gráfico

Exemplo	Resultado
`Upper(' abcD')`	Retorna 'ABCD'

Exemplo: Script de carregamento

```
Load  String,Upper(String) Inline [String rHode iSland washingTon d.C. new york];
```

Resultado

Cadeia	Upper(String)
rHode iSland	RHODE ISLAND
washingTon d.C.	WASHINGTON D.C.
new york	NEW YORK

5.25 Funções do sistema

As funções do sistema fornecem funções de acesso às Qlik Sense do sistema, dispositivo e aplicativos do Qlik Cloud.

Visão geral das funções do sistema

Algumas das funções são descritas adicionalmente após a visão geral. Para essas funções específicas, você pode clicar no nome da função na sintaxe para acessar imediatamente seus detalhes.

Author()
Esta função de script retorna uma string contendo a propriedade do autor do aplicativo atual. Pode ser usado no script de carga de dados e em um gráfico de expressão.

A propriedade do autor não pode ser definida na versão atual do Qlik Sense. Se você migrar um documento do QlikView, a propriedade do autor será retida.

ClientPlatform()

Esta função retorna a string de agente do usuário do navegador do cliente. Pode ser usado no script de carga de dados e em um gráfico de expressão.

Exemplo:

```
Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/35.0.1916.114 Safari/537.36
```

ComputerName

Esta função retorna uma string que contém o nome do computador, conforme retornado pelo sistema operacional. Pode ser usado no script de carga de dados e em um gráfico de expressão.

Se o nome do computador tiver mais de 15 caracteres, a cadeia conterá apenas os 15 primeiros caracteres.

```
ComputerName( )
```

DocumentName

Esta função retorna caracteres que contêm o nome do aplicativo Qlik Sense atual, sem caminho, mas com extensão. Pode ser usado no script de carga de dados e em um gráfico de expressão.

```
DocumentName( )
```

DocumentPath

Esta função retorna caracteres que contêm o caminho completo do aplicativo atual do Qlik Sense. Pode ser usado no script de carga de dados e em um gráfico de expressão.

```
DocumentPath( )
```

Esta função não é suportada no modo padrão. .

DocumentTitle

Esta função retorna caracteres que contêm o título do aplicativo atual do Qlik Sense. Pode ser usado no script de carga de dados e em um gráfico de expressão.

```
DocumentTitle( )
```

EngineVersion

Essa função retorna a versão completa do mecanismo Qlik Sense como uma cadeia.

```
EngineVersion ()
```

GetCollationLocale

Esta função retorna o nome de cultura do local de agrupamento que é utilizado. Se a variável CollationLocale não tiver sido definida, o local real da máquina do usuário é retornado.

```
GetCollationLocale( )
```

GetObjectField

GetObjectField() retorna o nome da dimensão. **Index** é um número inteiro opcional que indica a dimensão que deve ser retornada.

```
GetObjectField - função de gráfico([index])
```

GetRegistryString

Esta função retorna o valor de uma chave no registro do Windows. Pode ser usado no script de carga de dados e em um gráfico de expressão.

```
GetRegistryString(path, key)
```

Esta função não é suportada no modo padrão. .

IsPartialReload

Esta função retorna -1 (True) se a recarga atual for parcial; caso contrário, retorna 0 (False).

```
IsPartialReload ()
```

InObject

A função de gráfico **InObject()** avalia se o objeto atual está ou não contido em outro objeto com o ID especificado no argumento da função. O objeto pode ser uma pasta ou uma visualização.

```
InObject - função de gráfico(id_str)
```

ObjectId

A função de gráfico **ObjectId()** retorna o ID do objeto no qual a expressão é avaliada. A função usa um argumento opcional especificando a qual tipo de objeto a função diz respeito. O objeto pode ser uma pasta ou uma visualização. Essa função está disponível apenas em expressões de gráfico.

```
ObjectId - função de gráfico([object_type_str])
```

OSUser

Esta função retorna uma string que contém o nome do usuário conectado no momento. Pode ser usado no script de carga de dados e em um gráfico de expressão.

```
OSUser( )
```

No Qlik Sense Desktop e no Qlik Sense Mobile Client Managed, essa função sempre retorna 'Personal\Me'.

ProductVersion

Essa função retorna a versão completa do Qlik Sense e o número da compilação como uma string.

Essa função é obsoleta e substituída por **EngineVersion()**.

```
ProductVersion ()
```

ReloadTime
Esta função retorna uma data/hora de conclusão da última carga de dados. Pode ser usado no script de carga de dados e em um gráfico de expressão.

```
ReloadTime( )
```

StateName
StateName() retorna o nome do estado alternativo da visualização na qual ele é utilizado. StateName pode ser usado, por exemplo, para criar visualizações com texto e cores dinâmicos para refletir quando o estado de uma visualização é alterado. Essa função pode ser utilizada em expressões gráficas, mas não pode ser utilizada para determinar o estado ao qual a expressão se refere.

```
StateName - função de gráfico()
```

EngineVersion

Essa função retorna a versão completa do mecanismo Qlik Sense como uma cadeia.

Sintaxe:

```
EngineVersion()
```

InObject - função de gráfico

A função de gráfico **InObject()** avalia se o objeto atual está ou não contido em outro objeto com o ID especificado no argumento da função. O objeto pode ser uma pasta ou uma visualização.

Essa função pode ser usada para mostrar a hierarquia de objetos em uma pasta, desde o objeto de pasta de nível superior até visualizações aninhadas em outras visualizações. Essa função pode ser usada junto com as funções **if** e **ObjectId** para criar navegação personalizada em seus aplicativos.

Sintaxe:

```
InObject(id_str)
```

Tipo de dados de retorno: Booleano

No Qlik Sense, o valor booleano “true” é representado por -1, e o valor falso é representado por 0.

Argumentos

Argumento	Descrição
id_str	Um valor de string representando a ID do objeto que está sendo avaliado.

O ID da pasta pode ser obtido no URL do aplicativo. Para visualizações, use as opções de **Desenvolvedor** para identificar o ID do objeto e a string de texto do tipo de objeto.

Faça o seguinte:

1. No modo de análise, adicione o seguinte texto à sua URL:
   /options/developer
2. Clique com o botão direito do mouse em uma visualização e clique em **Desenvolvedor**.
3. Em **Propriedades**, obtenha o ID do objeto no cabeçalho do diálogo e o tipo de objeto na propriedade **"qType"**.

Limitações:

Essa função pode gerar resultados inesperados quando invocada em um objeto (por exemplo, um botão) dentro de um contêiner que é um item mestre. Essa limitação também se aplica aos itens mestres do painel de filtro, que são contêineres para várias caixas de listagem. Isso se deve à forma como os itens mestres usam a hierarquia de objetos.

InObject() é frequentemente usado em combinação com as seguintes funções:

Funções relacionadas

Função	Interação
if (page 569)	As funções **if** e **ObjectId** podem ser usadas juntas para criar expressões condicionais. Por exemplo, visualizações podem obter cores condicionais por meio de expressões usando essas funções.
ObjectId - função de gráfico (page 1490)	Semelhante a **if**, **ObjectID** também é usado com **InObject** para criar expressões condicionais.

Exemplo 1 – Funcionalidade básica

Expressão e resultados do gráfico

O exemplo básico a seguir demonstra como determinar se um objeto está contido em outro objeto. Nesse caso, verificaremos se um objeto de **Texto e imagem** reside em um objeto de pasta usando o ID da pasta como argumento.

Faça o seguinte:

1. Abra uma nova pasta e arraste um **gráfico** de Imagem e texto até a pasta.
2. No painel de propriedades, clique em **Adicionar medida**.
3. Clique em fx para abrir o editor de expressão.
4. Cole a seguinte expressão no diálogo:
   `=InObject()`
5. Modifique a expressão para incluir o ID da sua pasta como uma string entre parênteses.
   Por exemplo, para uma pasta com ID 1234-5678, você usaria o seguinte:
   `=InObject('1234-5678')`
6. Clique em **Aplicar**.

O valor `-1` é exibido no gráfico, indicando que a expressão foi avaliada como true.

Exemplo 2 – Objetos com cores condicionais

Expressão e resultados do gráfico

Visão geral

O exemplo a seguir demonstra como criar botões de navegação personalizados mostrando cores diferentes para indicar a pasta que está aberta no momento.

Comece criando um novo aplicativo e abrindo o Editor de carregamento de dados. Cole o script de carregamento a seguir em uma nova guia: Observe que os dados em si são um espaço reservado e não serão usados no conteúdo do exemplo.

Script de carregamento

```
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
```

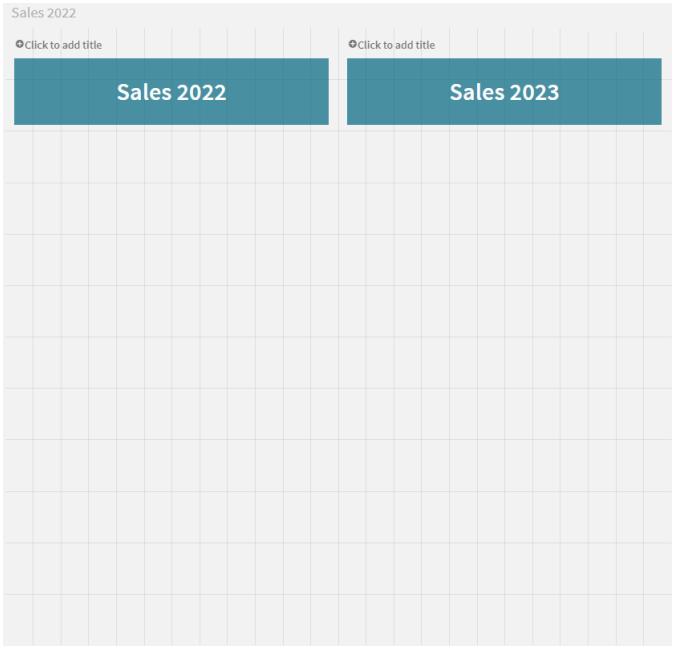
```
8194,'4/7/2022',40.39
8195,'5/16/2022',87.21
8196,'6/15/2022',95.93
8197,'7/26/2022',45.89
8198,'8/9/2022',36.23
8199,'9/22/2022',25.66
8200,'11/23/2022',82.77
8201,'12/27/2022',69.98
8202,'1/1/2023',76.11
8203,'2/8/2022',25.12
8204,'3/19/2022',46.23
8205,'6/26/2022',84.21
8206,'9/14/2022',96.24
8207,'11/29/2022',67.67
];
```

Criando as visualizações

Carregue os dados e crie duas novas pastas. Chame-os de *Sales 2022* e *Sales 2023*, respectivamente.

Em seguida, crie dois objetos de botão que serão usados para navegar entre as duas pastas.

Faça o seguinte:

1. Adicione dois objetos **Botão** à pasta.
2. Em **Aparência** > **Geral**, defina o **Rótulo** de cada botão como *Sales 2022* e *Sales 2023*, respectivamente.
3. Organize os botões para combinar com a imagem a seguir.
   Arranjo da pasta Sales 2022 com dois botões de navegação
   

4. Selecione o botão *Sales 2022* e expanda **Ações e navegação** no painel de propriedades.
5. Clique em **Adicionar ação** e, em **Navegação**, selecione **Ir para uma pasta**.

6. Em **Pasta**, selecione *Sales 2022*.
7. Repita a configuração da ação desse botão para vincular o botão **Sales 2023** à pasta *Sales 2023*.
8. Converta os botões em itens mestres clicando com o botão direito do mouse neles e selecionando **Adicionar a itens mestres**.

Agora, você pode copiar cada botão e colá-lo na pasta *Sales 2023*, usando o mesmo tamanho e disposição na pasta.

Criando cores condicionais

Em seguida, configure os botões para que fiquem azuis se estiverem vinculados à pasta aberta no momento e para que fiquem cinza claro se estiverem vinculados à pasta que não está aberta.

Faça o seguinte:

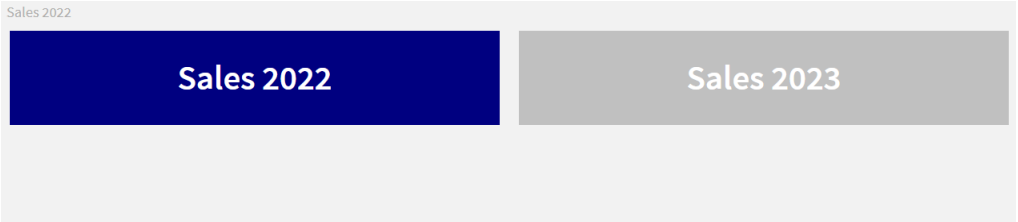
1. Abra a pasta *Sales 2022* e obtenha o ID dessa pasta no URL. Mantenha a pasta *Sales 2022* aberta.
2. Clique no **item mestre** do botão Sales 2022 e selecione **Editar** no painel de propriedades.
3. Em **Aparência** > **Plano de fundo**, selecione para colorir o botão **Por expressão**.
4. Em **Expressão**, cole o seguinte texto:
   =if(InObject(''), Blue(), LightGray())
5. Entre parênteses na expressão acima, cole o ID da pasta *Sales 2022*.

O botão agora está configurado para ficar azul quando a pasta *Sales 2022* estiver aberta e para ficar cinza claro quando não estiver aberta.

Repita as instruções acima para a pasta *Sales 2023*, vinculando o item mestre do botão **Sales 2023** ao ID da pasta *Sales 2023*.

Cada folha agora deve ter dois botões indicando a pasta aberta no momento com a cor azul.

Pasta Sales 2022 com coloração azul para indicar que Sales 2022 está sendo exibido atualmente



IsPartialReload

Esta função retorna -1 (True) se a recarga atual for parcial; caso contrário, retorna 0 (False).

Sintaxe:

```
IsPartialReload()
```

ObjectId - função de gráfico

A função de gráfico **ObjectId()** retorna o ID do objeto no qual a expressão é avaliada. A função usa um argumento opcional especificando a qual tipo de objeto a função diz respeito. O objeto pode ser uma pasta ou uma visualização. Essa função está disponível apenas em expressões de gráfico.

Sintaxe:

```
ObjectId([object_type_str])
```

Tipo de dados de retorno: string

O único argumento da função, **object_type_str**, é opcional e se refere a um valor de string representando o tipo do objeto.

Argumentos

Argumento	Descrição
object_type_str	Um valor de string representando o tipo do objeto que está sendo avaliado.

Se nenhum argumento for especificado na expressão da função, **ObjectId()** retornará o ID do objeto no qual a expressão é usada. Para retornar o ID do objeto de pasta no qual a visualização aparece, use *ObjectId('sheet')*.

No caso de objetos de visualização aninhados em outros objetos de visualização, especifique o tipo de objeto desejado no argumento da função para resultados diferentes. Por exemplo, para um **gráfico** de Imagem e texto em um contêiner, use *'text-image'* para retornar o objeto **Imagem e texto** e *'container'* para retornar o ID do contêiner.

Faça o seguinte:

1. No modo de análise, adicione o seguinte texto à sua URL:
   /options/developer
2. Clique com o botão direito do mouse em uma visualização e clique em **Desenvolvedor**.
3. Em **Propriedades**, obtenha o ID do objeto no cabeçalho do diálogo e o tipo de objeto na propriedade **"qType"**.

Limitações:

Essa função pode gerar resultados inesperados quando invocada em um objeto (por exemplo, um botão) dentro de um contêiner que é um item mestre. Essa limitação também se aplica aos itens mestres do painel de filtro, que são contêineres para várias caixas de listagem. Isso se deve à forma como os itens mestres usam a hierarquia de objetos.

A expressão de gráfico *ObjectId('sheet')* retornará uma string vazia nesses casos, enquanto *ObjectId ('masterobject')* mostrará o identificador do item mestre proprietário.

ObjectId() é frequentemente usado em combinação com as seguintes funções:

Funções relacionadas

Função	Interação
if (page 569)	As funções **if** e **ObjectId** podem ser usadas juntas para criar expressões condicionais. Por exemplo, visualizações podem obter cores condicionais por meio de expressões usando essas funções.
InObject - função de gráfico (page 1485)	Semelhante a **if**, **InObject** também é usado com **ObjectId** para criar expressões condicionais.

Exemplo 1 – Retornar o ID do objeto de gráfico

Expressão e resultados do gráfico

O exemplo básico a seguir demonstra como retornar o ID de uma visualização.

Faça o seguinte:

1. Abra uma nova pasta e arraste um gráfico de **Texto e imagem** até a pasta.
2. No painel de propriedades, clique em **Adicionar medida**.
3. Clique em fx para abrir o editor de expressão.
4. Cole a seguinte expressão no diálogo:
   `=ObjectId()`
5. Clique em **Aplicar**.

O ID do objeto de **Texto e imagem** é exibido na visualização.

O mesmo resultado pode ser obtido com a seguinte expressão:

```
=ObjectId('text-image')
```

Exemplo 2 – Retornar o ID da pasta

Expressão e resultados do gráfico

O exemplo básico a seguir demonstra como retornar o ID da pasta na qual uma visualização aparece.

Faça o seguinte:

1. Abra uma nova pasta e arraste um gráfico de **Texto e imagem** até a pasta.
2. No painel de propriedades, clique em **Adicionar medida**.
3. Clique em fx para abrir o editor de expressão.
4. Cole a seguinte expressão no diálogo:
   `=ObjectId('sheet')`
5. Clique em **Aplicar**.

O ID da pasta é exibido na visualização.

Exemplo 3 – Expressão aninhada

Expressão e resultados do gráfico

O exemplo a seguir mostra como a função **ObjectId()** pode ser aninhada dentro de outras expressões.

Faça o seguinte:

1. Abra uma nova pasta e arraste um gráfico de **Texto e imagem** até a pasta.
2. No painel de propriedades, clique em **Adicionar medida**.
3. Clique em fx para abrir o editor de expressão.
4. Cole a seguinte expressão no diálogo:
   =if(InObject(ObjectId('text-image')), 'In Text & image', 'Not in Text & image')
5. Clique em **Aplicar**.

O texto *In Text & image* aparece no gráfico, indicando que o objeto referenciado na expressão é um gráfico de **Texto e imagem**.

Para obter um exemplo mais detalhado usando cores condicionais, consulte o exemplo em *InObject - função de gráfico (page 1485)*.

ProductVersion

Essa função retorna a versão completa do Qlik Sense e o número da compilação como uma string. Essa função é obsoleta e substituída por **EngineVersion()**.

Sintaxe:

```
ProductVersion()
```

StateName - função de gráfico

StateName() retorna o nome do estado alternativo da visualização na qual ele é utilizado. StateName pode ser usado, por exemplo, para criar visualizações com texto e cores dinâmicos para refletir quando o estado de uma visualização é alterado. Essa função pode ser utilizada em expressões gráficas, mas não pode ser utilizada para determinar o estado ao qual a expressão se refere.

Sintaxe:

```
StateName ()
```

Example 1:

Texto dinâmico

```
='Region - ' & if(StateName() = '$', 'Default', StateName())
```

Example 2:

Cores dinâmicas

```
if(StateName() = 'Group 1', rgb(152, 171, 206),
      if(StateName() = 'Group 2', rgb(187, 200, 179),
            rgb(210, 210, 210)
      )
)
```

5.26 Funções de tabela

As funções de tabela retornam informações sobre a tabela de dados que está sendo lida no momento. Se nenhum nome de tabela for especificado e a função for usada em um comando **LOAD**, a tabela atual será usada.

Todas as funções podem ser usadas no script de carga de dados, enquanto que apenas **NoOfRows** pode ser usada em uma expressão do gráfico.

Visão geral das funções da tabela

Algumas das funções são descritas adicionalmente após a visão geral. Para essas funções específicas, você pode clicar no nome da função na sintaxe para acessar imediatamente seus detalhes.

FieldName

Essa **FieldName** função de script retorna o nome do campo com o número especificado dentro de uma tabela previamente carregada. Se a função for utilizada em um comando **LOAD**, ela não deverá fazer referência à tabela que estiver sendo carregada no momento.

```
FieldName (field_number ,table_name)
```

FieldNumber

Essa **FieldNumber** função de script retorna o número de um campo especificado em uma tabela previamente carregada. Se a função for utilizada em um comando **LOAD**, ela não deverá fazer referência à tabela que estiver sendo carregada no momento.

```
FieldNumber (field_name ,table_name)
```

NoOfFields

Essa **NoOfFields** função de script retorna o número de campos em uma tabela previamente carregada. Se a função for utilizada em um comando **LOAD**, ela não deverá fazer referência à tabela que estiver sendo carregada no momento.

```
NoOfFields (table_name)
```

NoOfRows

A função **NoOfRows** retorna o número de linhas (registros) em uma tabela anteriormente carregada. Se a função for utilizada em um comando **LOAD**, ela não deverá fazer referência à tabela que estiver sendo carregada no momento.

```
NoOfRows (table_name)
```

NoOfTables

Essa função de script retorna o número de tabelas carregadas anteriormente.

```
NoOfTables()
```

TableName

Essa função de script retorna o nome da tabela com o número especificado.

```
TableName(table_number)
```

TableNumber

Essa função de script retorna o número da tabela especificada. A primeira tabela tem o número 0.

Se table_name não existir, NULL é retornado.

```
TableNumber(table_name)
```

Exemplo:

Neste exemplo, queremos criar uma tabela com informações sobre as tabelas e campos que foram carregados.

Primeiro, carregamos alguns dados de amostra. Isso cria as duas tabelas que serão utilizadas para ilustrar as funções de tabela descritas nesta seção.

```
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 ;
```

Em seguida, vamos reiterar as tabelas que foram carregadas, usando a função **NoOfTables**, e, em seguida, através dos campos de cada tabela, utilizando a função **NoOfFields**, e carregamos as informações usando as funções da tabela.

```
//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;
```

A tabela resultante Tables ficará assim:

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

Essa **FieldName** função de script retorna o nome do campo com o número especificado dentro de uma tabela previamente carregada. Se a função for utilizada em um comando **LOAD**, ela não deverá fazer referência à tabela que estiver sendo carregada no momento.

Sintaxe:

```
FieldName(field_number ,table_name)
```

Argumentos:

Argumentos

Argumento	Descrição
field_number	O número de campo do campo que você quer referenciar.
table_name	A tabela contendo o campo que você quer referenciar.

Exemplo:

```
LET a = FieldName(4,'tab1');
```

FieldNumber

Essa **FieldNumber** função de script retorna o número de um campo especificado em uma tabela previamente carregada. Se a função for utilizada em um comando **LOAD**, ela não deverá fazer referência à tabela que estiver sendo carregada no momento.

Sintaxe:

```
FieldNumber(field_name ,table_name)
```

Argumentos:

Argumentos

Argumento	Descrição
field_name	É o nome do campo.
table_name	O nome da tabela contendo o campo.

Se o campo field_name não existe em table_name, ou table_name não existe, a função retorna 0.

Exemplo:

```
LET a = FieldNumber('Customer','tab1');
```

NoOfFields

Essa **NoOfFields** função de script retorna o número de campos em uma tabela previamente carregada. Se a função for utilizada em um comando **LOAD**, ela não deverá fazer referência à tabela que estiver sendo carregada no momento.

Sintaxe:

```
NoOfFields(table_name)
```

Argumentos:

Argumentos

Argumento	Descrição
table_name	O nome da tabela.

Exemplo:

```
LET a = NoOfFields('tab1');
```

NoOfRows

A função **NoOfRows** retorna o número de linhas (registros) em uma tabela anteriormente carregada. Se a função for utilizada em um comando **LOAD**, ela não deverá fazer referência à tabela que estiver sendo carregada no momento.

Sintaxe:

```
NoOfRows(table_name)
```

Argumentos:

Argumentos

Argumento	Descrição
table_name	O nome da tabela.

Exemplo:

```
LET a = NoOfRows('tab1');
```

5.27 Funções trigonométricas e hiperbólicas

Esta seção descreve as funções usadas para realizar operações trigonométricas e hiperbólicas. Em todas as funções, os argumentos são expressões resolvendo para ângulos listados em radianos, onde **x** deve ser interpretado como um número real.

Todos os ângulos são medidos em radianos.

Todas as funções podem ser usadas no script de carga de dados e em expressões do gráfico.

cos

Cosseno de **x**. O resultado é um número entre -1 e 1.

```
cos( x )
```

acos

Co-seno inverso de **x**. A função somente é definida se -1≤**x**≤1. O resultado é um número entre 0 e π.

```
acos( x )
```

sin
Seno de **x**. O resultado é um número entre -1 e 1.

```
sin( x )
```

asin
Seno inverso de **x**. A função somente é definida se -1≤**x**≤1. O resultado é um número entre -π/2 e π/2.

```
asin( x )
```

tan
Tangente de **x**. O resultado é um número real.

```
tan( x )
```

atan
Tangente inversa de**x**. O resultado é um número entre -π/2 e π/2.

```
atan( x )
```

atan2
Generalização bidimensional da função tangente inversa. Retorna o ângulo entre a origem e o ponto representado pelas coordenadas **x** e **y**. O resultado é um número entre - π e + π.

```
atan2( y,x )
```

cosh
Co-seno hiperbólico de **x**. O resultado é um número real positivo.

```
cosh( x )
```

sinh
Seno hiperbólico de **x**. O resultado é um número real.

```
sinh( x )
```

tanh
Tangente hiperbólica de **x**. O resultado é um número real.

```
tanh( x )
```

acosh
Inverter co-seno hiperbólico de **x**. O resultado é um número real positivo.

```
acosh( x )
```

asinh
Inverter seno hiperbólico de **x**. O resultado é um número real.

```
asinh( x )
```

atanh

Inverter tangente hiperbólica de **x**. O resultado é um número real.

```
atanh( x )
```

Exemplos:

O seguinte código de script carrega uma tabela de amostra e, depois, carrega uma tabela contendo as operações trigonométricas e hiperbólicas calculadas sobre os valores.

```
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;
```

6 Restrição de acesso do sistema de arquivo

Por razões de segurança, o Qlik Sense no modo padrão não oferece suporte a caminhos no script de carregamento de dados ou a funções e variáveis que expõem o sistema de arquivos.

No entanto, como caminhos do sistema de arquivos eram compatíveis no QlikView, é possível desabilitar o modo padrão e usar o modo legado para reutilizar scripts de carregamento do QlikView.

Desativar o modo padrão pode criar um risco de segurança, expondo o sistema de arquivo.

Desativando o modo padrão (page 1506)

6.1 Aspectos de segurança ao conectar-se com conexões de dados ODBC e OLE DB baseadas em arquivos

Conexões de dados ODBC e OLE DB usando drivers baseados em arquivo irá expor o caminho para o arquivo de dados conectado na sequência de conexão. O caminho pode ser exibido quando a conexão é editada, na caixa de diálogo de seleção de dados ou em certas consultas SQL. Esse é o caso com o modo padrão e o modo legado.

Se for uma preocupação expor o caminho do arquivo de dados, recomenda-se conectar-se ao arquivo de dados usando uma pasta de conexão de dados, se possível.

6.2 Limitações do modo padrão

Vários comandos, variáveis e funções não podem ser utilizados ou têm limitações no modo padrão. Usar comandos não suportados no script de carga de dados causa um erro quando o script de carga é executado. As mensagens de erro podem ser encontradas no arquivo de log do script. Usar variáveis e funções não suportadas não gera mensagens de erro ou entradas dearquivo de log. Em vez disso, a função retorna NULL.

Não há nenhuma indicação de que uma variável, comando ou função não seja suportado durante a edição do script de carga de dados.

Variáveis de sistema

Variáveis de sistema

Variável	Modo padrão	Modo legado	Definição
Floppy	Não suportado	Suportado	Retorna a letra da unidade do primeiro disco flexível encontrado, geralmente *a:*.

Variável	Modo padrão	Modo legado	Definição
CD	Não suportado	Suportado	Retorna a letra da primeira unidade de CD-ROM encontrada. Se nenhum CD-ROM for encontrado, *c:* será retornado.
QvPath	Não suportado	Suportado	Retorna os caracteres de busca do executável do Qlik Sense.
QvRoot	Não suportado	Suportado	Retorna o diretório raiz do executável do Qlik Sense.
QvWorkPath	Não suportado	Suportado	Retorna os caracteres de busca do aplicativo atual do Qlik Sense.
QvWorkRoot	Não suportado	Suportado	Retorna o diretório raiz do aplicativo atual do Qlik Sense.
WinPath	Não suportado	Suportado	Retorna os caracteres de busca para o Windows.
WinRoot	Não suportado	Suportado	Retorna o diretório raiz do Windows.
$(include=...)	Entrada com suporte: Caminho usando uma conexão da biblioteca	Entrada com suporte: Caminho usando uma conexão com a biblioteca ou o sistema de arquivos	A variável **Include/Must_Include** especifica um arquivo que contém texto, o qual deve ser incluído no script e avaliado como código de script. Ela não é usada para adicionar dados. Você pode armazenar partes do seu código de script em um arquivo de texto separado e reutilizá-lo em vários aplicativos. Essa é uma variável definida pelo usuário.

Comandos regulares de script

Comandos regulares de script

Comando	Modo padrão	Modo legado	Definição
Binary	Entrada com suporte: Caminho usando uma conexão da biblioteca	Entrada com suporte: Caminho usando uma conexão com a biblioteca ou o sistema de arquivos	O comando **binary** é usado para carregar dados a partir de outro aplicativo.
Connect	Entrada com suporte: Caminho usando uma conexão da biblioteca	Entrada com suporte: Caminho usando uma conexão com a biblioteca ou o sistema de arquivos	O comando **CONNECT** é usado para definir o acesso do Qlik Sense a uma base de dados geral por meio da interface OLE DB/ODBC. Para ODBC, a fonte de dados deve ser inicialmente especificada usando o administrador ODBC.
Directory	Entrada com suporte: Caminho usando uma conexão da biblioteca	Entrada com suporte: Caminho usando uma conexão com a biblioteca ou o sistema de arquivos	O comando **Directory** define o diretório para procurar os arquivos de dados em comandos **LOAD** subsequentes, até que um novo comando **Directory** seja feito.
Execute	Não suportado	Entrada com suporte: Caminho usando uma conexão com a biblioteca ou o sistema de arquivos	O comando **Execute** é utilizado para executar outros programas durante o carregamento de dados do Qlik Sense. Por exemplo, para fazer conversões que sejam necessárias.

Comando	Modo padrão	Modo legado	Definição
LOAD from ...	Entrada com suporte: Caminho usando uma conexão da biblioteca	Entrada com suporte: Caminho usando uma conexão com a biblioteca ou o sistema de arquivos	A declaração **LOAD** carrega campos de um arquivo, de dados definidos no script, de uma tabela de entrada carregada anteriormente, de uma página da Web, do resultado de um comando **SELECT** subsequente ou gerando dados automaticamente.
Store into ...	Entrada com suporte: Caminho usando uma conexão da biblioteca	Entrada com suporte: Caminho usando uma conexão com a biblioteca ou o sistema de arquivos	A instrução **Store** cria um arquivo QVD, Parquet, CSV ou TXT.

Comandos de controle de script

Comandos de controle de script

Comando	Modo padrão	Modo legado	Definição
For each... filelist mask/dirlist mask	Entrada com suporte: Caminho usando uma conexão da biblioteca Saída retornada: Conexão da biblioteca	Entrada com suporte: Caminho usando uma conexão com a biblioteca ou o sistema de arquivos Saída retornada: Caminho de conexão com a biblioteca ou do sistema de arquivos, dependendo da entrada	A sintaxe filelist mask produz uma lista separada por vírgulas de todos os arquivos existentes no diretório atual que correspondem à **filelist mask**. A sintaxe **dirlist mask** produz uma lista separada por vírgulas de todos os diretórios existentes no diretório atual que correspondem à máscara de nome do diretório.

Funções de arquivo

Funções de arquivo

Função	Modo padrão	Modo legado	Definição
Attribute()	Entrada com suporte: Caminho usando uma conexão da biblioteca	Entrada com suporte: Caminho usando uma conexão com a biblioteca ou o sistema de arquivos	Retorna o valor das meta tags de diferentes arquivos de mídia como texto.
ConnectString()	Saída retornada: Nome da conexão da biblioteca	Nome da conexão da biblioteca ou conexão real, dependendo da entrada	Retorna os caracteres de conexão habilitados para as conexões ODBC ou OLE DB.
FileDir()	Saída retornada: Conexão da biblioteca	Saída retornada: Caminho de conexão com a biblioteca ou do sistema de arquivos, dependendo da entrada	A função **FileDir** retorna uma string contendo o caminho do diretório do arquivo de tabela que está sendo lido no momento.
FilePath()	Saída retornada: Conexão da biblioteca	Saída retornada: Caminho de conexão com a biblioteca ou do sistema de arquivos, dependendo da entrada	A função **FilePath** retorna uma string contendo o caminho completo do arquivo de tabela que está sendo lido no momento.
FileSize()	Entrada com suporte: Caminho usando uma conexão da biblioteca	Entrada com suporte: Caminho usando uma conexão com a biblioteca ou o sistema de arquivos	A função **FileSize** retorna um inteiro contendo o tamanho em bytes do arquivo filename ou, se nenhum filename for especificado, retorna o do arquivo de tabela que está sendo lido no momento.

Função	Modo padrão	Modo legado	Definição
FileTime()	Entrada com suporte: Caminho usando uma conexão da biblioteca	Entrada com suporte: Caminho usando uma conexão com a biblioteca ou o sistema de arquivos	A função **FileTime** retorna um carimbo de data/hora no formato UTC da última modificação de um arquivo especificado. Se um arquivo não for especificado, a função retornará um carimbo de data/hora em UTC da última modificação do arquivo de tabela atualmente lido.
GetFolderPath()	Não suportado	Saída retornada: Caminho absoluto	A função **GetFolderPath** retorna o valor da função do Microsoft Windows *SHGetFolderPath*. Esta função admite como entrada o nome de uma pasta doMicrosoft Windows e retorna o caminho completo da pasta.
QvdCreateTime()	Entrada com suporte: Caminho usando uma conexão da biblioteca	Entrada com suporte: Caminho usando uma conexão com a biblioteca ou o sistema de arquivos	Esta função de script retorna o carimbo de data/hora do cabeçalho XML de um arquivo QVD, se houver, do contrário retorna NULL. No carimbo de data/hora, a hora é fornecida em UTC.
QvdFieldName()	Entrada com suporte: Caminho usando uma conexão da biblioteca	Entrada com suporte: Caminho usando uma conexão com a biblioteca ou o sistema de arquivos	Esta função de script retorna o nome do número do campo **fieldno** em um arquivo QVD. Se o campo não existir, será retornado NULL.

Função	Modo padrão	Modo legado	Definição
QvdNoOfFields()	Entrada com suporte: Caminho usando uma conexão da biblioteca	Entrada com suporte: Caminho usando uma conexão com a biblioteca ou o sistema de arquivos	Essa função de script retorna o número de campos em um arquivo QVD.
QvdNoOfRecords()	Entrada com suporte: Caminho usando uma conexão da biblioteca	Entrada com suporte: Caminho usando uma conexão com a biblioteca ou o sistema de arquivos	Essa função de script retorna o número de registros atualmente presentes em um arquivo QVD.
QvdTableName()	Entrada com suporte: Caminho usando uma conexão da biblioteca	Entrada com suporte: Caminho usando uma conexão com a biblioteca ou o sistema de arquivos	Essa função de script retorna o nome da tabela armazenada em um arquivo QVD.

Funções do sistema

Funções do sistema

Função	Modo padrão	Modo legado	Definição
DocumentPath()	Não suportado	Saída retornada: Caminho absoluto	Esta função retorna caracteres que contêm o caminho completo do aplicativo atual do Qlik Sense.
GetRegistryString()	Não suportado	Suportado	Retorna o valor de uma chave de registro nomeada com um determinado caminho de registro. Esta função pode ser utilizada em gráficos e scripts.

6.3 Desativando o modo padrão

É possível desativar o modo padrão ou, em outras palavras, definir o modo legado, para reutilizar os scripts de carga do QlikView que se referem a caminhos de arquivos absolutos ou relativos, bem como conexões da biblioteca.

Desativar o modo padrão pode criar um risco de segurança, expondo o sistema de arquivo.

Qlik Sense

Para o Qlik Sense, o modo padrão pode ser desativado no QMC usando a propriedade **Modo padrão**.

Qlik Sense Desktop

No Qlik Sense Desktop, é possível definir o modo padrão/legado em *Settings.ini*.

Se você tiver instalado o Qlik Sense Desktop usando o local de instalação padrão, *Settings.ini* estará localizado em *C:\Users\{user}\Documents\Qlik\Sense\Settings.ini*. Se você tiver instalado o Qlik Sense Desktop em uma pasta selecionada, *Settings.ini* estará localizado na pasta *Engine* do caminho de instalação.

Faça o seguinte:

1. Abra *Settings.ini* em um editor de texto.
2. Altere *StandardReload=1* para *StandardReload=0*.
3. Salve o arquivo e inicie o Qlik Sense Desktop.

O Qlik Sense Desktop agora é executado no modo legado.

Configurações

As definições disponíveis para StandardReload são:

- 1 (modo padrão)
- 0 (modo legado)

6 Scripts em nível de gráfico

Ao modificar dados do gráfico, você usa um subconjunto do script do Qlik Sense, que consiste em várias instruções. Um comando pode ser comum ou de controle. Alguns comandos podem ser precedidos por prefixos.

Comandos comuns geralmente são usados para manipular dados de uma forma ou de outra. Esses comandos podem ser escritos em qualquer quantidade de linhas no script e devem sempre ser encerrados por um ponto-e-vírgula ";".

Comandos de controle geralmente são utilizados para controlar o fluxo de execução do script. Cada cláusula de um comando de controle deve ser mantido dentro de uma linha do script e pode ser encerrada por um ponto e vírgula ou pelo fim de linha.

Os prefixos podem ser usados com comandos comuns aplicáveis, mas nunca com comandos de controle.

Todas as palavras-chave do script podem ser digitadas com qualquer combinação de caracteres maiúsculos e minúsculos. No entanto, os nomes de campos e de variáveis usados nos comandos diferenciam maiúsculas de minúsculas.

Nesta seção, você pode encontrar uma lista alfabética de todos os comandos de script, comandos de controle e prefixos disponíveis no subconjunto do script usado ao modificar dados de gráficos.

6.4 Comandos de controle

Ao modificar dados do gráfico, você usa um subconjunto do script do Qlik Sense, que consiste em várias instruções. Um comando pode ser comum ou de controle.

Comandos de controle geralmente são utilizados para controlar o fluxo de execução do script. Cada cláusula de um comando de controle deve ser mantido dentro de uma linha do script e pode ser encerrado por ponto e vírgula ou fim da linha.

Prefixos nunca são aplicados a comandos de controle.

Todas as palavras-chave do script podem ser digitadas com qualquer combinação de caracteres maiúsculos e minúsculos.

Visão geral de instruções de comando do modificador de gráfico

Cada função é descrita adicionalmente após a visão geral. Você também pode clicar no nome da função na sintaxe para acessar imediatamente os detalhes dessa função específica.

Call

O comando de controle **call** chama uma sub-rotina que deve ser definida por um comando **sub** prévio.

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

Do..loop

A declaração de controle **do..loop** é uma construção de iteração de script que executa um ou vários comandos até uma condição lógica ser atendida.

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

End

A palavra chave do script **End** é usada para fechar cláusulas **If**,**Sub** e **Switch**.

Exit

A palavra chave do script **Exit** é parte do comando **Exit Script**, mas também pode ser usada para sair das cláusulas **Do**, **For** ou **Sub**.

Exit script

Esse comando de controle interrompe a execução do script. Ele pode ser inserido em qualquer parte do script.

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

For..next

O comando de controle **for..next** cria uma construção de iteração de script com um contador. Os comandos dentro do loop incluídos entre **for** e **next** serão executados para cada valor da variável do contador, entre os limites inferior e superior especificados.

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

[statements]

[exit for [ ( when | unless ) condition ]

[statements]

Next [counter]
```

For each ..next

O comando de controle **for each..next** cria uma construção de iteração de script que executa um ou vários comandos para cada valor de uma lista separada por vírgulas. Os comandos dentro do loop incluídos entre **for** e **next** serão executados para cada valor da lista.

```
For each..next var in list

[statements]

[exit for [ ( when | unless ) condition ]

[statements]

next [var]
```

If..then

O comando de controle **if..then** é uma construção de seleção do script que força a execução do script seguir caminhos diferentes, dependendo de uma ou várias condições lógicas.

> *Como o comando **if..then** é um comando de controle e, portanto, termina com um ponto e vírgula ou com um fim de linha, cada uma das quatro cláusulas possíveis (**if..then**,**elseif..then**, **else** e **end if**) não deve ultrapassar o limite da linha.*

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

  [ statements ]

{ elseif condition then

  [ statements ] }

[ else

  [ statements ] ]

end if
```

Next

A palavra chave do script **Next** é usada para fechar loops **For**.

Sub

A declaração de controle **sub..end sub** define uma sub-rotina que pode ser acionada por meio de um comando **call**.

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

Switch

O comando de controle **switch** é uma construção de seleção do script que força a execução do script siga caminhos diferentes, dependendo do valor de uma expressão.

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

To

A palavra chave do script **To** é usada em vários comandos de script.

Call

O comando de controle **call** chama uma sub-rotina que deve ser definida por um comando **sub** prévio.

Sintaxe:

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

Argumentos:

Argumentos

Argumento	Descrição
name	O nome da sub-rotina.
paramlist	Uma lista separada por vírgulas dos parâmetros reais que serão enviados para a sub-rotina. Cada item da lista deve ser um nome de campo, uma variável ou uma expressão arbitrária.

A sub-rotina chamada por uma declaração **call** deve ser definida por um **sub** encontrado anteriormente durante a execução do script.

Os parâmetros são copiados na sub-rotina e, se o parâmetro no comando **call** for uma variável e não uma expressão, serão copiados novamente ao sair da sub-rotina.

Limitações:

- Como o comando **call** é de controle e, portanto, termina com um ponto e vírgula ou com um fim de linha, ele não deve cruzar um limite de linha.
- Quando você define uma sub-rotina com `Sub..end sub` dentro de um comando de controle, por exemplo `if..then`, você só pode chamar a sub-rotina de dentro do mesmo comando de controle.

Do..loop

A declaração de controle **do..loop** é uma construção de iteração de script que executa um ou vários comandos até uma condição lógica ser atendida.

Sintaxe:

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

*Como o comando **do..loop** é um comando de controle e, portanto, termina com um ponto e vírgula ou com um fim de linha, cada uma das três cláusulas possíveis (**do**,**exit do** e **loop**) não deve ultrapassar o limite da linha.*

Argumentos:

Argumentos

Argumento	Descrição
condition	Uma expressão lógica de avaliação como True ou False.

Argumento	Descrição
statements	Qualquer grupo de um ou mais comandos de script do Qlik Sense.
while / until	A cláusula condicional **while** ou **until** deve aparecer apenas uma vez em qualquer declaração **do..loop**, isto é, depois de **do** ou depois de **loop**. Cada condição é interpretada somente na primeira ocorrência, mas é avaliada em todas as outras ocorrências no loop.
exit do	Se uma cláusula **exit do** for encontrada dentro do loop, a execução do script será transferida para o primeiro comando após a cláusula **loop**, indicando o fim do loop. Uma cláusula **exit do** pode ser transformada em condicional pelo uso opcional de um sufixo **when** ou **unless**.

End

A palavra chave do script **End** é usada para fechar cláusulas **If**,**Sub** e **Switch**.

Exit

A palavra chave do script **Exit** é parte do comando **Exit Script**, mas também pode ser usada para sair das cláusulas **Do**, **For** ou **Sub**.

Exit script

Esse comando de controle interrompe a execução do script. Ele pode ser inserido em qualquer parte do script.

Sintaxe:

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

Como o comando **exit script** é de controle e, portanto, termina com um ponto e vírgula ou com um fim de linha, ele não deve cruzar um limite de linha.

Argumentos:

Argumentos

Argumento	Descrição
condition	Uma expressão lógica de avaliação como True ou False.
when / unless	Um comando **exit script** pode ser transformado em condicional pelo uso opcional de um sufixo **when** ou **unless**.

Exemplos:

```
//Exit script
Exit Script;

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

For..next

O comando de controle **for..next** cria uma construção de iteração de script com um contador. Os comandos dentro do loop incluídos entre **for** e **next** serão executados para cada valor da variável do contador, entre os limites inferior e superior especificados.

Sintaxe:

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

[statements]

[exit for [ ( when | unless ) condition ]

[statements]

Next [counter]
```

As expressões *expr1*, *expr2* e *expr3* serão avaliadas somente na primeira vez que o loop é inserido. O valor da variável counter pode ser alterado por comandos dentro da referência circular, entretanto, essa não é uma prática de programação recomendável.

Se uma cláusula **exit for** for encontrada dentro do loop, a execução do script será transferida para o primeiro comando após a cláusula **next**, indicando o fim do loop. Uma cláusula **exit for** pode ser transformada em condicional pelo uso opcional de um sufixo **when** ou **unless**.

*Como o comando **for..next** é um comando de controle e, portanto, termina com um ponto e vírgula ou com um fim de linha, cada uma das três cláusulas possíveis (**for..to..step**,**exit for** e **next**) não deve ultrapassar o limite da linha.*

Argumentos:

Argumentos

Argumento	Descrição
counter	Um nome de variável. Se *counter* for especificado depois de **next**, ele deverá ter o mesmo nome de variável que a encontrada após o **for** correspondente.
expr1	Uma expressão que determina o primeiro valor da variável *counter* para o qual o loop deve ser executado.
expr2	Uma expressão que determina o último valor da variável *counter* para o qual o loop deve ser executado.
expr3	Uma expressão que determina o valor que indica o incremento da variável *counter* cada vez que o loop é executado.

Argumento	Descrição
condition	uma expressão lógica de avaliação como True ou False.
statements	Qualquer grupo de um ou mais comandos de script do Qlik Sense.

For each..next

O comando de controle **for each..next** cria uma construção de iteração de script que executa um ou vários comandos para cada valor de uma lista separada por vírgulas. Os comandos dentro do loop incluídos entre **for** e **next** serão executados para cada valor da lista.

Sintaxe:
A sintaxe especial permite gerar listas com nomes de arquivo e diretório no diretório atual.

```
for each var in list

[statements]

[exit for [ ( when | unless ) condition ]

[statements]

next [var]
```

Argumentos:

Argumentos

Argumento	Descrição
var	Um nome de uma variável de script que adquire um novo valor da lista para cada execução de referência de loop. Se **var** for especificado depois de **next**, ele deverá ter o mesmo nome de variável que a encontrada após o **for each** correspondente.

O valor da variável **var** pode ser alterado por comandos dentro da referência circular, entretanto, essa não é uma prática de programação recomendável.

Se uma cláusula **exit for** for encontrada dentro do loop, a execução do script será transferida para o primeiro comando após a cláusula **next**, indicando o fim do loop. Uma cláusula **exit for** pode ser transformada em condicional pelo uso opcional de um sufixo **when** ou **unless**.

*Como o comando **for each..next** é um comando de controle e, portanto, termina com um ponto e vírgula ou com um fim de linha, cada uma das três cláusulas possíveis (**for each**,**exit for** e **next**) não deve ultrapassar o limite da linha.*

Sintaxe:

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

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

Argumentos

Argumento	Descrição
constant	Qualquer número ou string. Observe que uma string escrita diretamente no script deve ser colocada entre aspas simples. Uma string sem aspas simples será interpretada como uma variável e, em seguida, o valor da variável será usado. Os números não precisam ser colocados entre aspas simples.
expression	Uma expressão arbitrária.
mask	A máscara de um nome de arquivo ou de um nome de pasta que pode incluir todos os caracteres válidos de nome de arquivo, bem como os caracteres curingas padrão ***** e **?**. Você pode usar caminhos de arquivo absolutos ou caminhos lib://.
condition	Uma expressão lógica de avaliação como True ou False.
statements	Qualquer grupo de um ou mais comandos de script do Qlik Sense.
filelist mask	Essa sintaxe produz uma lista separada por vírgulas de todos os arquivos existentes no diretório atual e que correspondem à máscara de nome de arquivo. *Este argumento suporta apenas as conexões de dados da biblioteca no modo padrão.*
dirlist mask	Essa sintaxe produz uma lista separada por vírgulas de todas as pastas existentes na pasta atual e que correspondem à máscara de nome de pasta. *Este argumento suporta apenas as conexões de dados da biblioteca no modo padrão.*
fieldvaluelist mask	Essa sintaxe itera através dos valores de um campo já carregado no Qlik Sense.

*O Qlik Conectores do provedor de armazenamento na Web e outras conexões de DataFiles não oferecem suporte para máscaras de filtro que usam os caracteres curinga (***** e **?**).*

Example 1: Carregando uma lista de arquivos

```
// 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: Criando uma lista de arquivos no disco

Este exemplo carrega uma lista de todos os arquivos do Qlik Sense relacionados em uma pasta.

```
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: Iterações através dos valores de um campo

Este exemplo itera através da lista dos valores carregados do FIELD e gera um novo campo, NEWFIELD. Para cada valor do FIELD, será criado dois registros NEWFIELD.

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

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

A tabela resultante tem a seguinte aparência:

Example table

NEWFIELD
one-1
one-2

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

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

O comando de controle **if..then** é uma construção de seleção do script que força a execução do script seguir caminhos diferentes, dependendo de uma ou várias condições lógicas.

Comandos de controle geralmente são utilizados para controlar o fluxo de execução do script. Em uma expressão de gráfico, use a função condicional **if** em vez disso.

Sintaxe:

```
If  condition then

  [ statements ]

{ elseif condition then

  [ statements ] }

[ else

  [ statements ] ]

end if
```

Como o comando **if..then** é um comando de controle e, portanto, termina com um ponto e vírgula ou com um fim de linha, cada uma das quatro cláusulas possíveis (**if..then**,**elseif..then**, **else** e **end if**) não deve ultrapassar o limite da linha.

Argumentos:

Argumentos

Argumento	Descrição
condition	Uma expressão lógica que pode ser avaliada como True ou False.
statements	Qualquer grupo de um ou mais comandos de script do Qlik Sense.

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

A palavra chave do script **Next** é usada para fechar loops **For**.

Sub..end sub

A declaração de controle **sub..end sub** define uma sub-rotina que pode ser acionada por meio de um comando **call**.

Sintaxe:

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

Argumentos são copiados na sub-rotina e, se os parâmetros reais correspondentes na instrução **call** forem o nome de uma variável, eles serão copiados novamente após a saída da sub-rotina.

Se uma sub-rotina tiver mais parâmetros formais que os parâmetros reais transmitidos por um comando **call**, os parâmetros extra serão inicializados como NULL e poderão ser utilizados como variáveis locais na sub-rotina.

Argumentos:

Argumentos

Argumento	Descrição
name	O nome da sub-rotina.
paramlist	Uma lista, separada por vírgulas, de nomes de variáveis para os parâmetros formais da sub-rotina. Eles podem ser usados como qualquer variável dentro da sub-rotina.
statements	Qualquer grupo de um ou mais comandos de script do Qlik Sense.

Limitações:

- Como o comando **sub** é um comando de controle e, portanto, termina com um ponto e vírgula ou com um fim de linha, cada uma de suas duas cláusulas possíveis (**sub** e **end sub**) não deve ultrapassar o limite da linha.
- Quando você define uma sub-rotina com `Sub..end sub` dentro de um comando de controle, por exemplo `if..then`, você só pode chamar a sub-rotina de dentro do mesmo comando de controle.

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: – transferência de parâmetro

```
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)
```

O resultado do exibido acima será local, dentro da sub-rotina, A será inicializado para 1, B será inicializado para 4 e C será inicializado para NULL.

Durante a saída da sub-rotina, a variável global A receberá 2 como valor (copiado da sub-rotina). O segundo parâmetro real “(X+1)*2” não será copiado de volta porque não é uma variável. Por fim, a variável global C não será afetada pela chamada de sub-rotina.

Switch..case..default..end switch

O comando de controle **switch** é uma construção de seleção do script que força a execução do script siga caminhos diferentes, dependendo do valor de uma expressão.

Sintaxe:

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

> *Como o comando **switch** é um comando de controle e, portanto, termina com um ponto e vírgula ou com um fim de linha, cada uma das quatro cláusulas possíveis (**switch**,**case**, **default** e **end switch**) não deve ultrapassar o limite da linha.*

Argumentos:

Argumentos

Argumento	Descrição
expression	Uma expressão arbitrária.
valuelist	Uma lista de valores separados por vírgulas com os quais o valor da expressão será comparado. A execução do script continuará com os comandos do primeiro grupo encontrado com um valor em valuelist igual ao valor da expressão. Cada valor da valuelist pode ser uma expressão arbitrária. Se nenhuma correspondência for encontrada em nenhuma cláusula **case**, as declarações da cláusula **default**, se especificadas, serão executadas.
statements	Qualquer grupo de um ou mais comandos de script do Qlik Sense.

Exemplo:

```
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

A palavra chave do script **To** é usada em vários comandos de script.

6.5 Prefixos

Os prefixos podem ser usados com comandos comuns aplicáveis, mas nunca com comandos de controle.

Todas as palavras-chave do script podem ser digitadas com qualquer combinação de caracteres maiúsculos e minúsculos. No entanto, os nomes de campos e de variáveis usados nos comandos diferenciam maiúsculas de minúsculas.

Visão geral de prefixos de modificadores de gráfico

Cada função é descrita adicionalmente após a visão geral. Você também pode clicar no nome da função na sintaxe para acessar imediatamente os detalhes dessa função específica.

Add
O prefixo **Add** pode ser incluído em qualquer comando **LOAD** ou **SELECT** no script para especificar que deve adicionar registros a outra tabela. Ele também especifica que esse comando deve ser executado em um carregamento parcial. O prefixo **Add** também pode ser usado em um comando **Map**.

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

Replace
O prefixo **Replace** pode ser adicionado a qualquer comando **LOAD** ou **SELECT** no script para especificar que a tabela carregada deve substituir outra tabela. Ele também especifica que esse comando deve ser executado em um carregamento parcial. O prefixo **Replace** também pode ser usado em um comando **Map**.

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

Add

Em um contexto de modificação de gráfico, o prefixo **Add** é usado com **LOAD** para anexar valores à tabela *HC1*, representando o hipercubo calculado pelo Qlik associative engine. Você pode especificar uma ou várias colunas. Os valores ausentes são preenchidos automaticamente pelo Qlik associative engine.

Sintaxe:

```
Add loadstatement
```

Exemplo:

Este exemplo adiciona duas linhas às colunas *Datas* e *Vendas* a partir do comando inline

```
Add Load
x as Dates,
y as Sales
Inline
[
Dates,Sales
2001/09/1,1000
```

```
2001/09/10,-300
]
```

Replace

Em um contexto de modificação de gráfico, o prefixo **Replace** altera todos os valores da tabela *HC1* com um valor calculado definido pelo script.

Sintaxe:

```
Replace loadstatement
```

Exemplo:

Este exemplo substitui todos os valores na coluna `z` pela soma de `x` e `y`.

```
Replace Load
x+y as z
Resident HC1;
```

6.6 Comandos regulares

Comandos comuns geralmente são usados para manipular dados de uma forma ou de outra. Esses comandos podem ser escritos em qualquer quantidade de linhas no script e devem sempre ser encerrados por um ponto-e-vírgula ";".

Todas as palavras-chave do script podem ser digitadas com qualquer combinação de caracteres maiúsculos e minúsculos. No entanto, os nomes de campos e de variáveis usados nos comandos diferenciam maiúsculas de minúsculas.

Visão geral de comandos regulares de modificadores de gráfico

Cada função é descrita adicionalmente após a visão geral. Você também pode clicar no nome da função na sintaxe para acessar imediatamente os detalhes dessa função específica.

LOAD

Em um contexto de modificação de gráfico, a instrução **LOAD** carrega dados adicionais no hipercubo a partir de dados definidos no script ou de uma tabela anteriormente carregada. Também é possível carregar dados de conexões analíticas.

*O comando **LOAD** deve ter um prefixo **Replace** ou **Add**, ou será rejeitado.*

```
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
O comando **let** é um complemento ao comando **set**, usado para definir variáveis de script. O comando **let**, ao contrário do comando **set**, avalia a expressão no lado direito do sinal de igual "=" no tempo de execução do script antes de ser atribuída à variável.

```
Let variablename=expression
```

Set
O comando **set** é usado para definir as variáveis do script. Essas variáveis podem ser utilizadas para substituir strings, caminhos, unidades e assim por diante.

```
Set variablename=string
```

Put
O comando **Put** é usado para definir algum valor numérico no hipercubo.

HCValue
A instrução **HCValue** é usada para recuperar valores em uma linha de uma coluna especificada.

Load

Em um contexto de modificação de gráfico, a instrução **LOAD** carrega dados adicionais no hipercubo a partir de dados definidos no script ou de uma tabela anteriormente carregada. Também é possível carregar dados de conexões analíticas.

*O comando **LOAD** deve ter um prefixo **Replace** ou **Add**, ou será rejeitado.*

Sintaxe:

```
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 ]
```

Argumentos:

Argumentos

Argumento	Descrição
fieldlist	*fieldlist ::= (* | field{, * | field })* Uma lista dos campos a serem carregados. O uso de * como lista de campos indica todos os campos da tabela. *field ::= (fieldref | expression) [***as** *aliasname]* A definição de campo deve conter sempre um literal, uma referência a um campo existente ou a uma expressão. *fieldref ::= (fieldname |@fieldnumber |@startpos:endpos [* ***I*** | ***U*** | ***R*** | ***B*** | **T**]) *fieldname* é um texto idêntico a um nome de campo da tabela. Observe que o nome do campo deverá estar entre aspas duplas ou colchetes se contiver espaços, por exemplo. Às vezes, os nomes de campo não estão explicitamente disponíveis. Assim, uma notação diferente será usada: *@fieldnumber* representa o número de campo em um arquivo de tabela delimitado. Deve ser um inteiro positivo, precedido de “@”. A numeração sempre inicia no número 1 até o número de campos. *@startpos*:*endpos* representa as posições inicial e final de um campo em um arquivo com registros de comprimento fixo. As posições devem ser inteiros positivos. Os dois números devem ser precedidos de "@" e separados por dois-pontos. A numeração sempre inicia no número 1 até o número de posições. No último campo, **n** é usado como posição final do campo. • Se *@startpos*:*endpos* for seguido imediatamente pelos caracteres **I** ou **U**, os bytes lidos serão interpretados como um inteiro binário assinado (**I**) ou não assinado (**U**) (ordem dos bytes da Intel). O número das posições lidas deve ser 1, 2 ou 4. • Se *@startpos*:*endpos* for imediatamente seguido pelo caractere **R**, os bytes lidos serão interpretados como um número real binário (ponto flutuante IEEE de 32 ou de 64 bits). O número das posições lidas deve ser 4 ou 8. • Se *@startpos*:*endpos* for seguido imediatamente pelo caractere **B**, os bytes lidos serão interpretados como números BCD (Binary Coded Decimal), de acordo com o padrão COMP-3. Qualquer número de bytes pode ser especificado. *expression* pode ser uma função numérica ou uma função de string baseada em um ou vários outros campos da mesma tabela. Para obter mais informações, consulte a sintaxe das expressões. **as** é usado para atribuir um novo nome ao campo.

Argumento	Descrição
inline	**inline** será usado se os dados precisarem ser digitados no script, e não carregados de um arquivo. *data ::= [text]* Os dados inseridos por uma cláusula **inline** devem ser colocados entre aspas duplas ou colchetes. O texto entre esses elementos será interpretado da mesma maneira que o conteúdo de um arquivo. Portanto, no local em que uma nova linha for inserida em um arquivo de texto, você também deverá inseri-la no texto de uma cláusula **inline**, isto é, pressionando a tecla Enter ao digitar o script. O número de colunas é definido pela primeira linha. *format-spec ::= (fspec-item {, fspec-item })* A especificação do formato consiste em uma lista de vários itens de especificação de formato, entre colchetes. Para obter mais informações, consulte *Itens de especificação de formato (page 167)*.
resident	**resident** é usado se for necessário carregar dados de uma tabela anteriormente carregada. *table label* é um rótulo que precede o comando **LOAD** que criou a tabela original. O rótulo deve ter dois-pontos no final.

Argumento	Descrição
extension	É possível carregar dados de conexões analíticas. Você precisa usar a cláusula **extension** para chamar uma função definida no plug-in SSE (Server-Side Extension) ou avaliar um script. Você pode enviar uma única tabela ao plug-in SSE, e uma única tabela de dados é retornada. Se o plug-in não especificar os nomes dos campos que são retornados, os campos serão nomeados como Field1, Field2, e assim por diante. `Extension pluginname.functionname( tabledescription );` • Carregando dados com o uso de uma função em um plug-in SSE *tabledescription ::= (table { ,tablefield})* Se você não indicar campos de tabela, os campos serão usados na ordem de carregamento. • Carregando dados ao avaliar um script em um plug-in SSE *tabledescription ::= (script, table { ,tablefield})* **Manipulação de tipos de dados na definição de campo de tabela** Tipos de dados são detectados automaticamente em conexões analíticas. Se os dados não tiverem valores numéricos e pelo menos uma string de texto não NULL, o campo será considerado texto. Em qualquer outro caso, ele será considerado numérico. Você pode forçar o tipo de dados envolvendo um nome de campo com **String()** ou **Mixed()**. • **String()** força o campo como texto. Se o campo for numérico, a parte de texto do valor duplo será extraída, e nenhuma conversão será realizada. • **Mixed()** força o campo como duplo. **String()** ou **Mixed()** não podem ser usados fora de definições de campo de tabela **extension**, e você não pode usar outras funções Qlik Sense em uma definição de campo de tabela.
where	**where** é uma cláusula utilizada para declarar se um registro deve ou não ser incluído na seleção. A seleção será incluída se *criterion* for True. *criterion* é uma expressão lógica.
while	**while** é uma cláusula usada para declarar que um registro deve ser lido repetidamente. O mesmo registro será lido se *criterion* forTrue. Para que possa ser útil, a cláusula **while** normalmente deve incluir a função **IterNo( )**. *criterion* é uma expressão lógica.

Argumento	Descrição
group by	**group by** é uma cláusula usada para definir os campos de agregação (agrupamento) dos dados. Os campos de agregação devem ser incluídos de alguma maneira nas expressões carregadas. Nenhum outro campo diferente desses poderá ser usado fora das funções de agregação nas expressões carregadas. *groupbyfieldlist ::= (fieldname { ,fieldname })*
order by	**order by** é uma cláusula usada para classificar os registros de uma tabela residente antes de seu processamento pela declaração **load**. A tabela residente pode ser classificada por um ou mais campos, em ordem ascendente ou descendente. A classificação é feita, principalmente, pelo valor numérico e, em seguida, pela ordem nacional de colação. Essa cláusula somente poderá ser usada quando a fonte de dados for uma tabela residente. Os campos de classificação especificam por qual campo a tabela residente é classificada. O campo pode ser especificado por seu nome ou número na tabela residente (o primeiro campo é o número 1). orderbyfieldlist ::= fieldname [sortorder] { , fieldname [sortorder] } *sortorder* é *asc* para ascendente ou *desc* para descendente. Se nenhuma *sortorder* for especificada, *asc* será assumido. *fieldname*, *path*, *filename* e *aliasname* são strings de texto que indicam o que seus respectivos nomes implicitamente representam. Qualquer campo na tabela de origem pode ser usado como *fieldname*. No entanto, campos criados por meio da cláusula (*aliasname*) estão fora de escopo e não podem ser usados dentro de uma mesma declaração **load**.

Let

O comando **let** é um complemento ao comando **set**, usado para definir variáveis de script. O comando **let**, ao contrário do comando **set**, avalia a expressão no lado direito do sinal de igual "=" no tempo de execução do script antes de ser atribuída à variável.

Sintaxe:

```
Let variablename=expression
```

Exemplos e resultados:

Exemplo	Resultado

`Set x=3+4;`
 	
`Let y=3+4;`
 	
`z=$(y)+1;`
	`$(x)` será avaliado como ' 3+4 '
	
	`$(y)` será avaliado como ' 7 '
	
	`$(z)` será avaliado como ' 8 '
	
	Observe a diferença entre os comandos **Set** e **Let**. A instrução **Set** atribui a string "3+4" à variável, enquanto a instrução **Let** avalia a string e atribui 7 à variável.

Let T=now( );	`$(T)` receberá o valor da hora atual.

Set

O comando **set** é usado para definir as variáveis do script. Essas variáveis podem ser utilizadas para substituir strings, caminhos, unidades e assim por diante.

Sintaxe:

```
Set variablename=string
```

Example 1:

```
Set FileToUse=Data1.csv;
```

Example 2:

```
Set Constant="My string";
```

Example 3:

```
Set BudgetYear=2012;
```

Put

O comando **put** é usado para definir algum valor numérico no hipercubo.

O acesso às colunas pode ser feito por rótulos. Você também pode acessar colunas e linhas por ordem de declaração. Veja os exemplos abaixo para obter mais detalhes.

Sintaxe:

```
put column(position)=value
```

Example 1:

O acesso às colunas pode ser feito por rótulos.

Este exemplo definirá um valor de `1` na primeira posição da coluna rotulada *Sales*.

```
Put Sales(1) = 1;
```

Example 2:

Você pode acessar colunas de medidas por ordem de declaração usando o formato `#hc1.measure` para medidas.

Este exemplo definirá o valor 1000 na décima posição do hipercubo classificado como final.

```
Put #hc1.measure.2(10) = 1000;
```

Example 3:

Você pode acessar as linhas de dimensão por ordem de declaração usando o formato `#hc1.dimension` para dimensões.

Este exemplo coloca o valor da constante Pi na quinta linha da terceira dimensão declarada.

```
Put #hc1.dimension.3(5) = Pi();
```

Se não houver tais dimensões ou expressões, em valor ou rótulos, um erro será retornado indicando que a coluna não foi encontrada. Se o índice da coluna estiver fora dos limites, nenhum erro será exibido.

HCValue

A função **HCValue** é usada para recuperar valores em uma linha de uma coluna especificada.

Sintaxe:

```
HCValue(column,position)
```

Example 1:

Este exemplo retorna o valor na primeira posição da coluna com o rótulo “Sales”.

```
HCValue(Sales,1)
```

Example 2:

Este exemplo retorna o valor na décima posição do hipercubo classificado.

```
HCValue(#hc1.measure2,10)
```

Example 3:

Este exemplo retorna o valor na quinta linha na terceira dimensão.

```
HCValue(#hc1.dimension.3,5)
```

Se não houver tais dimensões ou expressões, em valor ou rótulos, um erro será retornado indicando que a coluna não foi encontrada. Se o índice da coluna estiver fora dos limites, NULL será retornado.

7 Funções e comandos do QlikView não suportados em Qlik Sense

A maioria das funções e comandos que podem ser usados em scripts de carga e expressões de gráficos QlikView também são suportados em Qlik Sense, mas existem algumas exceções, descritas aqui.

7.1 Comandos de script não suportados em Qlik Sense

Instruções de script QlikView sem suporte no Qlik Sense

Comando	Comentários
Command	No lugar, use **SQL**.
InputField	

7.2 Funções não suportadas em Qlik Sense

Esta lista descreve funções de gráficos e script do QlikView que não são suportados em Qlik Sense.

- **GetCurrentField**
- **GetExtendedProperty**
- **Input**
- **InputAvg**
- **InputSum**
- **MsgBox**
- **NoOfReports**
- **ReportComment**
- **ReportId**
- **ReportName**
- **ReportNumber**

7.3 Prefixos não suportados no Qlik Sense

Esta lista descreve prefixos do QlikView que não são suportados no Qlik Sense.

- **Bundle**
- **Image_Size**
- **Info**

8 Funções e comandos não recomendados em Qlik Sense

A maioria das funções e comandos que podem ser utilizados em scripts de carga e expressões de gráfico QlikView também são suportados em Qlik Sense, mas alguns deles não são recomendados para uso em Qlik Sense. Também há funções e comandos disponíveis em versões anteriores do Qlik Sense que ficaram obsoletas.

Por motivos de compatibilidade eles ainda funcionarão como planejado, mas é recomendável atualizar o código de acordo com as recomendações nesta seção, uma vez que podem ser removidos nas versões futuras.

8.1 Comandos de script não recomendados em Qlik Sense

Esta tabela contém instruções de script não recomendadas para uso no Qlik Sense.

Instruções de script não recomendadas

Comando	**Recomendação**
Command	No lugar, use **SQL**.
CustomConnect	No lugar, use **Custom Connect**.

8.2 Parâmetros de comandos de script não recomendados em Qlik Sense

Esta tabela descreve parâmetros de instruções de script não recomendados para uso no Qlik Sense.

Parâmetros de instruções de script não recomendados

Comando	**Parâmetros**
Buffer	Use **Incremental** no lugar de: • **Inc** (não recomendado) • **Incr** (não recomendado)

Comando	Parâmetros
LOAD	As palavras-chave do parâmetro a seguir são geradas pelos assistentes de transformação de arquivo QlikView. A funcionalidade é retida com a execução de script dos dados, mas Qlik Sense não fornece suporte/assistentes guiados para gerar um comando com esses parâmetros. • **Bottom** • **Cellvalue** • **Col** • **Colmatch** • **Colsplit** • **Colxtr** • **Compound** • **Contain** • **Equal** • **Every** • **Expand** • **Filters** • **Intarray** • **Interpret** • **Length** • **Longer** • **Numerical** • **Pos** • **Remove** • **Rotate** • **Row** • **Rowcnd** • **Shorter**

8.3 Funções não recomendadas em Qlik Sense

Esta tabela descreve funções de gráfico e de script não recomendadas para uso no Qlik Sense.

Funções não recomendadas

Função	Recomendação
NumAvg **NumCount** **NumMax** **NumMin** **NumSum**	No lugar, use funções de intervalo. *Funções de intervalo (page 1338)*
Color() **QliktechBlue** **QliktechGray**	Em vez disso, use outras funções de cor. É possível substituir **QliktechBlue()** por **RGB(8, 18, 90)** e **QliktechGray** por **RGB(158, 148, 137)** para obter as mesmas cores. *Funções de cor (page 558)*
QlikViewVersion	No lugar, use **EngineVersion**. *EngineVersion (page 1485)*
ProductVersion	No lugar, use **EngineVersion**. *EngineVersion (page 1485)*
QVUser	
Year2Date	No lugar, use **YearToDate**.
Vrank	No lugar, use **Rank**.
WildMatch5	No lugar, use **WildMatch**.

Qualificador **ALL**

No QlikView, o qualificador **ALL** pode ocorrer antes de uma expressão. Isso é equivalente a usar **{1} TOTAL**. Nesse caso, o cálculo será feito em todos os valores do campo no documento, ignorando as dimensões do gráfico e as seleções atuais. O mesmo valor é sempre retornado independentemente do estado lógico no documento. Se o qualificador **ALL** for usado, uma expressão de conjunto não poderá ser usada, visto que o qualificador **ALL** define um conjunto por si só. Por motivos de legado, o qualificador **ALL** continuará funcionando nessa versão do Qlik Sense, mas pode ser removido em versões futuras.