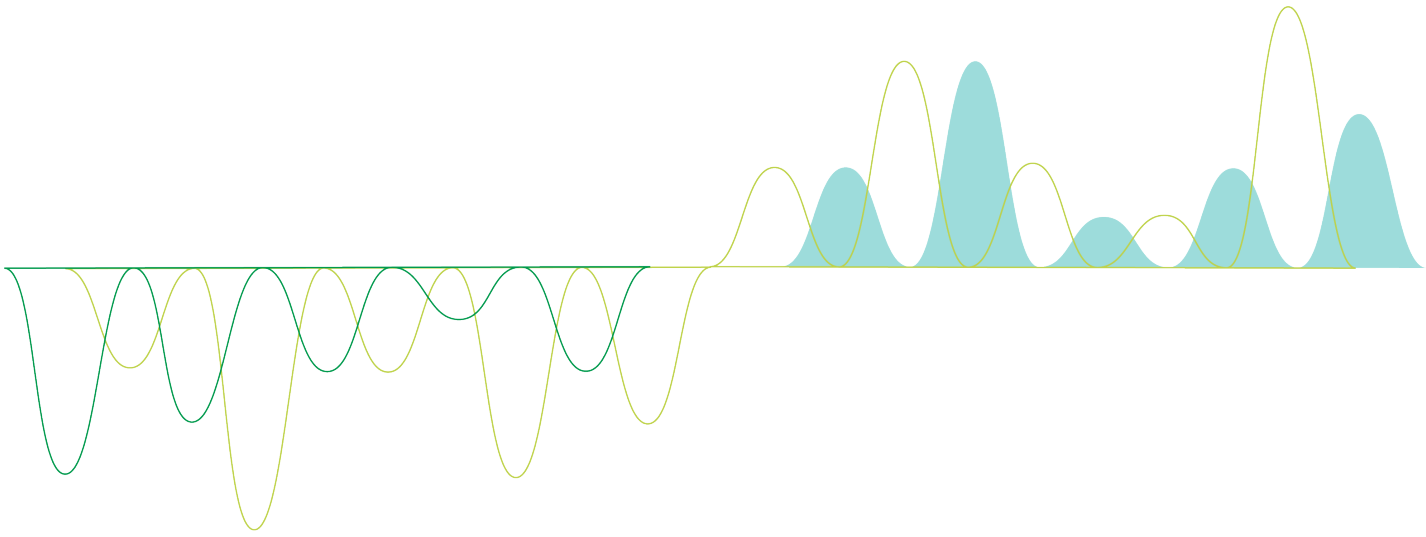
스크립트 구문 및 차트 함수

Qlik Sense®
February 2024



Contents

1 Qlik Sense는 무엇입니까?

Qlik Sense는 데이터 분석 플랫폼입니다. Qlik Sense를 사용하면 데이터를 분석하고 자신만의 데이터 검색을 만들 수 있습니다. 그룹 및 조직 전체에서 지식을 공유하고 데이터를 분석할 수 있습니다. Qlik Sense를 통해 스스로 질문을 하고 응답하며 자체적인 통찰력을 키울 수 있습니다. Qlik Sense는 여러분이 동료와 협력하여 결론에 도달하도록 지원합니다.

1.1 Qlik Sense에서 수행할 수 있는 작업

대부분의 BI(Business Intelligence) 제품은 문제를 미리 파악하여 해답을 낼 수 있도록 도와줍니다. 그렇다면 그 이후에 나오는 의문 사항은 어떻게 해야 할까요? 누군가 사용자의 보고서를 읽거나 시각화를 본 후 질문을 한다면 어떻게 하시겠습니까? Qlik Sense 연관 경험을 통해 자신만의 통찰력을 발휘하면서 연달아 이어지는 질문에 답할 수 있습니다. Qlik Sense를 사용하면 단 몇 번의 클릭만으로도 한 단계씩 배우고 이전의 결과를 기반으로 다음 단계를 진행하며 자유롭게 데이터를 탐색할 수 있습니다.

1.2 Qlik Sense 작동 방법

Qlik Sense는 즉석에서 정보 보기를 생성합니다. Qlik Sense에서는 사전 정의된 정적 보고서가 필요하거나 다른 사용자에게 의존할 필요가 없습니다. 클릭하고 배우기만 하면 됩니다. 클릭할 때마다 Qlik Sense가 즉시 응답하여 앱 내의 모든 Qlik Sense 시각화 및 보기를 사용자의 특정 선택에 맞게 새로 계산된 데이터 셋으로 업데이트합니다.

앱 모델

대규모 비즈니스 응용 프로그램을 배포 및 관리하는 대신, 재사용, 수정 및 다른 사용자와 공유가 가능한 자체 Qlik Sense 앱을 만들 수 있습니다. 앱 모델은 전문가에게 새로운 보고서 또는 시각화에 대해 문의하지 않고도 자체적으로 다음 질문에 대해 답을 하도록 도와줍니다.

연관 경험

Qlik Sense에서는 데이터 내 모든 관계를 자동으로 관리하고 **green/white/gray** 표시를 사용하여 정보를 나타냅니다. 선택은 녹색, 연관 데이터는 흰색, 제외된(연관되지 않은) 데이터는 회색으로 표시됩니다. 이러한 즉각적인 피드백을 통해 새로운 질문을 생각하고 계속 탐색 및 발견할 수 있습니다.

공동 작업 및 이동성

또한 Qlik Sense를 사용하면 때와 장소에 관계없이 동료들과 공동 작업을 수행할 수 있습니다. 연관 경험 및 공동 작업을 포함한 모든 Qlik Sense 기능은 모바일 장치에서 사용할 수 있습니다. Qlik Sense를 사용하여 어디에서나 동료들과 교류하며 묻고 답하고 질문에 대한 후속 조치를 취할 수 있습니다.

1.3 Qlik Sense 배포 방법

Qlik Sense는 Qlik Sense Desktop 및 Qlik Sense Enterprise의 두 가지 버전으로 배포할 수 있습니다.

Qlik Sense Desktop

이 버전은 일반적으로 로컬 컴퓨터에 설치하는 단일 사용자용 간편 설치 버전입니다.

Qlik Sense Enterprise

이 버전은 Qlik Sense 사이트를 배포하는 데 사용됩니다. 사이트는 공통의 논리적 리포지토리 또는 중앙 노드에 연결된 1대 이상의 서버 컴퓨터의 모음입니다.

1.4 Qlik Sense 사이트 관리 방법

Qlik 관리 콘솔를 사용하면 간편하고 직관적인 방식으로 Qlik Sense 사이트를 구성, 관리 및 모니터링할 수 있습니다. 라이센스, 액세스 및 보안 규칙을 관리하고 노드 및 데이터 소스 연결을 구성하고 기타 수많은 활동 및 리소스 간에 콘텐츠와 사용자를 구성할 수 있습니다.

1.5 Qlik Sense 확장 및 사용자에게 맞게 수정

Qlik Sense는 사용자가 자신만의 확장 기능을 개발하고 다음과 같은 다양한 용도로 Qlik Sense를 수정 및 통합할 수 있는 유연한 API 및 SDK를 제공합니다.

확장 기능 및 매시업 생성

여기서는 이제 JavaScript를 사용한 웹 개발을 수행하여 Qlik Sense 앱의 사용자 지정 시각화 확장 기능을 생성하거나, 매시업 API를 사용하여 Qlik Sense 콘텐츠로 웹 사이트를 구축합니다.

클라이언트 생성

.NET에서 클라이언트를 구축하고 자체 응용 프로그램에 Qlik Sense 개체를 포함할 수 있습니다. 또한 Qlik Sense 클라이언트 프로토콜을 사용하면 WebSocket 통신을 처리할 수 있는 프로그래밍 언어로 네이티브 클라이언트를 구축할 수도 있습니다.

서버 도구 생성

서비스 및 사용자 디렉터리 API를 사용하면 Qlik Sense 사이트를 관리하는 자체 도구를 생성할 수 있습니다.

다른 데이터 소스에 연결

Qlik Sense 커넥터를 생성하여 사용자 지정 데이터 소스에서 데이터를 검색합니다.

2 스크립트 구문 개요

2.1 스크립트 구문에 대한 소개

스크립트에는 논리에 포함된 데이터 소스의 이름, 테이블의 이름 및 필드의 이름이 정의됩니다. 또한 액세스 권한 정의 내의 필드도 스크립트에 정의됩니다. 스크립트는 연속적으로 실행되는 다수의 문으로 구성됩니다.

Qlik Sense 명령줄 구문과 스크립트 구문에는 Backus-Naur 이론(또는 BNF 코드)이라는 표기법이 사용됩니다.

코드의 첫 번째 줄은 새로운 Qlik Sense 파일을 만들 때 생성됩니다. 이러한 숫자 해석 변수의 기본값은 OS의 국가별 설정에 따라 유추됩니다.

스크립트는 연속적으로 실행되는 다수의 스크립트 문과 키워드로 구성됩니다. 모든 스크립트 문은 세미콜론("**;**")으로 끝나야 합니다.

LOAD 문에서 표현식 및 함수를 사용하면 로드된 데이터를 변환할 수 있습니다.

쉼표, 탭 또는 세미콜론이 구분 기호로 사용된 테이블 파일의 경우 **LOAD** 문을 사용할 수 있습니다. 기본적으로 **LOAD** 문은 파일의 모든 필드를 로드합니다.

ODBC 또는 OLE DB 데이터베이스 커넥터를 통해 일반 데이터베이스에 액세스할 수 있습니다. 여기에는 표준 SQL 문이 사용됩니다. 허용되는 SQL 구문은 서로 다른 ODBC 드라이버 간에 다릅니다.

또한 사용자 지정 커넥터를 사용하여 다른 데이터 소스에 액세스할 수도 있습니다.

2.2 Backus-Naur 이론이란?

Qlik Sense 명령줄 구문과 스크립트 구문에는 Backus-Naur 이론(BNF 코드로도 알려짐)이라는 표기법이 사용됩니다.

다음 표에는 BNF 코드에서 사용되는 기호 목록이 해석 방법에 대한 설명과 함께 나와 있습니다.

기호

기호	**설명**
|	논리적 OR: 어느 쪽의 기호든 사용할 수 있습니다.
( )	우선 순위를 정의하는 괄호: BNF 구문을 구조화하는 데 사용됩니다.
[ ]	대괄호: 둘러싸인 항목은 옵션입니다.
{ }	중괄호: 둘러싸인 항목은 0회 이상 반복될 수 있습니다.
기호	비종결 구문 범주: 다른 기호로 더 세분화하여 구분할 수 있습니다. 예를 들어 위 기호의 조합, 기타 비종결 기호, 텍스트 문자열 등이 있습니다.
::=	기호를 정의하는 블록의 시작을 표시합니다.
LOAD	텍스트 문자열로 구성된 종결 기호입니다. 있는 그대로 스크립트에 입력해야 합니다.

모든 종결 기호는 **bold face** 글꼴로 표시됩니다. 예를 들어 "("은 우선 순위를 정의하는 괄호로 해석해야 하며, "**(**"은 스크립트에 표시할 문자로 해석해야 합니다.

Alias 문에 대한 설명은 다음과 같습니다.

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

이는 뒤에 임의의 필드 이름, 텍스트 문자열 "as", 임의의 별칭 이름이 오는 텍스트 문자열 "alias"로 해석해야 합니다. 쉼표로 구분된 "fieldname as alias"의 추가적인 조합을 원하는 수대로 지정할 수 있습니다.

다음은 올바른 문의 형태입니다.

```
alias a as first;

alias a as first, b as second;

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

다음 문은 올바르지 않습니다.

```
alias a as first b as second;

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

3 스크립트 문 및 키워드

Qlik Sense 스크립트는 다수의 문으로 구성됩니다. 하나의 문은 정규 스크립트 문 또는 스크립트 제어 문일 수 있습니다. 특정 문은 접두사가 선행될 수 있습니다.

정규 문은 일반적으로 어떤 방식으로든 데이터를 편집하는 데 사용됩니다. 이러한 문은 스크립트에서 줄 수에 관계없이 작성할 수 있으며, 항상 세미콜론(";")으로 종결되어야 합니다.

제어 문은 일반적으로 스크립트 실행 흐름을 제어하는 데 사용됩니다. 제어 문의 각 절은 하나의 스크립트 줄 안에 유지되어야 하며 세미콜론 또는 줄의 끝으로 종결될 수 있습니다.

접두사는 해당되는 정규 문에는 적용할 수 있지만 제어 문에는 적용할 수 없습니다. 하지만 **when** 및 **unless** 접두사는 소수의 특정 제어 문 절에서 접미사로 사용할 수 있습니다.

다음 섹션에 모든 스크립트 문, 제어 문 및 접두사가 사전순으로 나열된 목록이 있습니다.

모든 스크립트 키워드는 소문자와 대문자를 원하는 대로 조합하여 입력할 수 있습니다. 하지만 문에 사용되는 필드 및 변수 이름은 대/소문자가 구분됩니다.

3.1 스크립트 제어 문

Qlik Sense 스크립트는 다수의 문으로 구성됩니다. 하나의 문은 정규 스크립트 문 또는 스크립트 제어 문일 수 있습니다.

제어 문은 일반적으로 스크립트 실행 흐름을 제어하는 데 사용됩니다. 제어 문의 각 절은 하나의 스크립트 줄 안에 유지되어야 하며 세미콜론 또는 줄의 끝으로 종결될 수 있습니다.

제어 문에는 접두사가 적용되지 않지만 예외적으로 소수의 특정 제어 문에는 **when** 및 **unless** 접두사를 사용할 수 있습니다.

모든 스크립트 키워드는 소문자와 대문자를 원하는 대로 조합하여 입력할 수 있습니다.

스크립트 제어 문 개요

각 함수는 개요가 끝난 후에 더 자세히 설명합니다. 구문에서 함수 이름을 클릭하여 해당 함수에 대한 상세 설명에 즉시 액세스할 수도 있습니다.

Call
call 제어 문은 앞에 **sub** 문으로 정의된 서브루틴을 호출합니다.

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

Do..loop
do..loop 제어 문은 논리 조건이 충족될 때까지 하나 또는 여러 문을 실행하는 스크립트 반복 구조입니다.

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

Exit script
이 제어 문은 스크립트 실행을 중지합니다. 스크립트 어느 곳에나 삽입할 수 있습니다.

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

For each ..next
for each..next 제어 문은 쉼표로 구분된 목록의 각 값을 대상으로 하나 또는 여러 문을 실행하는 스크립트 반복 구조입니다. 목록의 각 값에 대해 **for**와 **next**로 묶인 루프 내의 문이 실행됩니다.

```
For each..next var in list

[statements]

[exit for [ ( when | unless ) condition ]

[statements]

next [var]
```

For..next
for..next 제어 문은 카운터를 사용하는 스크립트 반복 구조입니다. 지정된 하한 및 상한 사이의 각 카운터 변수 값에 대해 **for**와 **next**로 묶인 루프 내의 문이 실행됩니다.

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

[statements]

[exit for [ ( when | unless ) condition ]

[statements]

Next [counter]
```

If..then
if..then 제어 문은 하나 또는 여러 논리 조건에 따라 스크립트 실행을 다른 경로로 전환하는 스크립트 선택 구조입니다.

> ***if..then** 문은 제어 문이므로 세미콜론이나 줄 끝(EOL)으로 끝나며 여기에 포함될 수 있는 4개의 절(**if..then, elseif..then, else** 및 **end if**)은 줄 경계를 넘지 않아야 합니다.*

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

  [ statements ]

{ elseif condition then

  [ statements ] }

[ else
```

```
  [ statements ] ]
```

```
end if
```

Sub
sub..end sub 제어 문은 **call** 문에서 호출할 수 있는 서브루틴을 정의합니다.

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

Switch
switch 제어 문은 표현식의 값에 따라 스크립트 실행을 다른 경로로 전환하는 스크립트 선택 구조입니다.

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

Call

call 제어 문은 앞에 **sub** 문으로 정의된 서브루틴을 호출합니다.

구문:

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

인수:

인수

인수	설명
name	서브루틴의 이름입니다.
paramlist	서브루틴으로 전달할 실제 매개 변수의 쉼표로 구분된 목록입니다. 목록의 각 항목은 필드 이름, 변수 또는 임의의 표현식이 될 수 있습니다.

call 문으로 호출하는 서브루틴은 스크립트 실행 시 초반에 발견되는 **sub** 문으로 정의되어 있어야 합니다.

매개 변수는 서브루틴으로 복사되며, **call** 문에 있는 매개 변수가 변수이고 표현식이 아니라면 서브루틴이 종료될 때 다시 역으로 복사됩니다.

제한 사항:

- **call** 문은 제어 문이고 세미콜론 또는 줄 끝 중 하나로 끝나므로 줄 경계를 넘어가지 말아야 합니다.
- 제어 문(예: `if..then`) 내에서 `sub..end sub`를 사용하여 서브루틴을 정의하면 동일한 제어 문 내에서만 서브루틴을 호출할 수 있습니다.

이 예는 폴더 및 하위 폴더의 모든 Qlik 관련 파일을 나열하고 테이블에 파일 정보를 저장합니다. 폴더에 대해 Apps라는 이름의 데이터 연결을 만들었다고 가정합니다.

DoDir 서브루틴은 폴더 'lib://Apps'에 대한 참조를 통해 매개 변수로 호출됩니다. 서브루틴 내에는 재귀 호출 `Call DoDir (Dir)`이 있으며 이 재귀 호출을 통해 함수가 하위 폴더에서 재귀적으로 파일을 찾을 수 있습니다.

```
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

do..loop 제어 문은 논리 조건이 충족될 때까지 하나 또는 여러 문을 실행하는 스크립트 반복 구조입니다.

구문:

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

> ***do..loop** 문은 제어 문이므로 세미콜론이나 줄 끝(EOL)으로 끝나며 여기에 포함될 수 있는 세 절 (**do**,**exit do** 및 **loop**)은 줄 경계를 넘지 않아야 합니다.*

인수:

인수

인수	설명
condition	True 또는 False로 평가되는 논리 표현식입니다.
statements	하나 이상의 Qlik Sense 스크립트 문의 그룹입니다.
while / until	**while** 또는 **until** 조건절은 **do..loop** 문에서 한 번만 나와야 합니다. 즉, **do** 다음이나 **loop** 다음에 나올 수 있습니다. 각 조건은 처음 발견될 때만 해석되지만 루프 내에서는 발견될 때마다 평가됩니다.
exit do	루프 내에서 **exit do** 절이 나올 경우 스크립트 실행이 루프의 끝을 나타내는 **loop** 절 다음 첫 번째 문으로 전환됩니다. 선택적으로 **when** 또는 **unless** 접미사를 사용하여 **exit do** 절을 조건부로 만들 수 있습니다.

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

Set a=1;

Do while a<10

LOAD * from file$(a).csv;

Let a=a+1;

Loop
```

End

End 스크립트 키워드는 **If**, **Sub** 및 **Switch** 절을 닫는 데 사용됩니다.

Exit

Exit 스크립트 키워드는 **Exit Script** 문에 속하지만 **Do**, **For** 또는 **Sub** 절을 종료할 때도 사용할 수 있습니다.

Exit script

이 제어 문은 스크립트 실행을 중지합니다. 스크립트 어느 곳에나 삽입할 수 있습니다.

구문:

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

exit script 문은 제어 문이고 세미콜론 또는 줄 끝 중 하나로 끝나므로 줄 경계를 넘어가지 말아야 합니다.

인수:

인수

인수	설명
condition	True 또는 False로 평가되는 논리 표현식입니다.
when / unless	선택적으로 **when** 또는 **unless** 절을 사용하여 **exit script** 문을 조건부로 만들 수 있습니다.

```
//Exit script
Exit Script;

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

For..next

for..next 제어 문은 카운터를 사용하는 스크립트 반복 구조입니다. 지정된 하한 및 상한 사이의 각 카운터 변수 값에 대해 **for**와 **next**로 묶인 루프 내의 문이 실행됩니다.

구문:

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

[statements]

[exit for [ ( when | unless ) condition ]

[statements]

Next [counter]
```

expr1, *expr2* 및 *expr3* 표현식은 루프에 처음 진입할 때만 평가됩니다. counter 변수의 값은 루프 내의 문에서 변경할 수 있지만 이는 좋은 프로그래밍 방법이 아닙니다.

루프 내에서 **exit for** 절이 나올 경우 스크립트 실행이 루프의 끝을 나타내는 **next** 절 다음 첫 번째 문으로 전환됩니다. 선택적으로 **when** 또는 **unless** 접미사를 사용하여 **exit for** 절을 조건부로 만들 수 있습니다.

> ***for..next** 문은 제어 문이므로 세미콜론이나 줄 끝(EOL)으로 끝나며 여기에 포함될 수 있는 세 절 (**for..to..step**,**exit for** 및 **next**)은 줄 경계를 넘지 않아야 합니다.*

인수:

인수

인수	설명
counter	변수 이름입니다. *counter*가 **next** 다음에 지정된 경우, 이 이름은 해당하는 **for** 다음에 오는 이름과 동일해야 합니다.
expr1	루프를 실행할 *counter* 변수의 첫 번째 값을 결정하는 표현식입니다.
expr2	루프를 실행할 *counter* 변수의 마지막 값을 결정하는 표현식입니다.
expr3	루프가 실행될 때마다 *counter* 변수의 증가분을 나타내는 값을 결정하는 표현식입니다.
condition	True 또는 False로 평가되는 논리 표현식입니다.
statements	하나 이상의 Qlik Sense 스크립트 문의 그룹입니다.

Example 1: 파일 시퀀스 로드

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

for a=1 to 9
```

```
	LOAD * from file$(a).csv;
next
```

Example 2: 임의의 숫자의 파일 로드

이 예에서는 데이터 파일 *x1.csv, x3.csv, x5.csv, x7.csv* 및 *x9.csv*가 있다고 가정합니다. 로드는 `if rand( )<0.5 then` 조건을 사용하여 임의의 지점에서 중지됩니다.

```
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

for each..next 제어 문은 쉼표로 구분된 목록의 각 값을 대상으로 하나 또는 여러 문을 실행하는 스크립트 반복 구조입니다. 목록의 각 값에 대해 **for**와 **next**로 묶인 루프 내의 문이 실행됩니다.

구문:
특수한 구문을 사용하면 현재 디렉터리 내의 파일과 디렉터리 이름으로 목록을 생성할 수 있습니다.

```
for each var in list

[statements]

[exit for [ ( when | unless ) condition ]

[statements]

next [var]
```

인수:

인수

인수	설명
var	각 루프 실행에 대해 목록에서 새 값을 가져오는 스크립트 변수 sname. **var**가 **next** 다음에 지정된 경우, 이 이름은 해당하는 **for each** 다음에 오는 이름과 동일해야 합니다.

var 변수의 값은 루프 내의 문에서 변경할 수 있지만 이것은 좋은 프로그래밍 방법이 아닙니다.

루프 내에서 **exit for** 절이 나올 경우 스크립트 실행이 루프의 끝을 나타내는 **next** 절 다음 첫 번째 문으로 전환됩니다. 선택적으로 **when** 또는 **unless** 접미사를 사용하여 **exit for** 절을 조건부로 만들 수 있습니다.

> ***for each..next** 문은 제어 문이므로 세미콜론이나 줄 끝(EOL)으로 끝나며 여기에 포함될 수 있는 세 절(**for each**,**exit for** 및 **next**)은 줄 경계를 넘지 않아야 합니다.*

구문:

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

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

인수

인수	설명
constant	임의의 숫자 또는 문자열입니다. 스크립트에 직접 포함되는 문자열은 작은따옴표로 묶어야 합니다. 작은따옴표가 없는 문자열은 변수로 해석되며 변수의 값이 사용됩니다. 숫자는 작은따옴표로 묶을 필요가 없습니다.
expression	임의의 표현식입니다.
mask	표준 와일드카드 문자 ***** 및 **?**를 비롯한 유효한 파일 이름 문자를 포함할 수 있는 파일 또는 디렉터리 이름 마스크입니다. 절대 파일 경로 또는 lib:// 경로를 사용할 수 있습니다.
condition	True 또는 False로 평가되는 논리 표현식입니다.
statements	하나 이상의 Qlik Sense 스크립트 문의 그룹입니다.
filelist mask	이 구문을 사용하면 현재 디렉터리에서 파일 이름 마스크와 일치하는 모든 파일의 쉼표로 구분된 목록이 생성됩니다. *이 인수는 표준 모드의 라이브러리 연결만 지원합니다.*
dirlist mask	이 구문을 사용하면 현재 폴더에서 폴더 이름 마스크와 일치하는 모든 폴더의 쉼표로 구분된 목록이 생성됩니다. *이 인수는 표준 모드의 라이브러리 연결만 지원합니다.*
fieldvaluelist mask	이 구문은 Qlik Sense로 이미 로드된 필드 값을 통해 반복합니다.

> *Qlik 웹 저장소 공급자 커넥터 및 기타 DataFiles 연결은 와일드카드(***** 및 **?**) 문자를 사용하는 필터 마스크를 지원하지 않습니다.*

Example 1: 파일 목록 로드

```
// 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: 디스크의 파일 목록 만들기

이 예에서는 폴더에 있는 모든 Qlik Sense 관련 파일 목록을 로드합니다.

```
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: 필드 값 a를 통해 반복

이 예에서는 로드된 FIELD 값의 목록을 통해 반복하고 새 필드인 NEWFIELD를 생성합니다. FIELD의 각 값에 대해서는 두 NEWFIELD 레코드가 생성됩니다.

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

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

결과 테이블은 다음과 같습니다.

Example table

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

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

if..then 제어 문은 하나 또는 여러 논리 조건에 따라 스크립트 실행을 다른 경로로 전환하는 스크립트 선택 구조입니다.

제어 문은 일반적으로 스크립트 실행 흐름을 제어하는 데 사용됩니다. 차트 표현식에서는 **if** 조건부 함수를 대신 사용합니다.

구문:

```
If  condition then

  [ statements ]

{ elseif condition then

  [ statements ] }

[ else

  [ statements ] ]

end if
```

if..then 문은 제어 문이므로 세미콜론이나 줄 끝(EOL)으로 끝나며 여기에 포함될 수 있는 4개의 절(**if..then**, **elseif..then**, **else** 및 **end if**)은 줄 경계를 넘지 않아야 합니다.

인수:

인수

인수	설명
condition	True 또는 False로 평가되는 논리 표현식입니다.
statements	하나 이상의 Qlik Sense 스크립트 문의 그룹입니다.

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

Next 스크립트 키워드는 **For** 루프를 닫는 데 사용됩니다.

Sub..end sub

sub..end sub 제어 문은 **call** 문에서 호출할 수 있는 서브루틴을 정의합니다.

구문:

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

인수는 서브루틴에 복사되며, **call** 문에 있는 해당 실제 매개 변수가 변수 이름일 경우 서브루틴을 종료할 때 다시 복제됩니다.

서브루틴에 **call** 문에 의해 전달된 실제 매개 변수의 수보다 더 많은 공식 매개 변수가 있는 경우 추가 매개 변수는 NULL로 초기화되며 서브루틴 내에서 로컬 변수로 사용될 수 있습니다.

인수:

인수

인수	설명
name	서브루틴의 이름입니다.

인수	설명
paramlist	서브루틴의 공식 매개 변수에 대한 쉼표로 구분된 변수 이름 목록입니다. 이들은 서브루틴 내의 어떠한 변수로도 사용할 수 있습니다.
statements	하나 이상의 Qlik Sense 스크립트 문의 그룹입니다.

제한 사항:

- **sub** 문은 제어 문이므로 세미콜론이나 줄 끝(EOL)으로 끝나며 여기에 포함될 수 있는 두 절(**sub** 및 **end sub**)은 줄 경계를 넘지 않아야 합니다.
- 제어 문(예: `if..then`) 내에서 `Sub..end sub`를 사용하여 서브루틴을 정의하면 동일한 제어 문 내에서만 서브루틴을 호출할 수 있습니다.

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: - 매개 변수 전달

```
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)
```

상기 예의 결과는 서브루틴 내에서 로컬로 이루어지며, A는 1로, B는 4로, C는 NULL로 초기화됩니다.

서브루틴을 종료할 때 전역 변수 A의 값은 2가 됩니다(서브루틴에서 다시 복사됨). 두 번째 실제 매개 변수 “(X+1)*2”는 변수가 아니기 때문에 다시 복사되지 않습니다. 마지막으로 전역 변수 C는 서브루틴 호출에 의해 영향을 받지 않습니다.

Switch..case..default..end switch

switch 제어 문은 표현식의 값에 따라 스크립트 실행을 다른 경로로 전환하는 스크립트 선택 구조입니다.

구문:

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

> ***switch** 문은 제어 문이므로 세미콜론이나 줄 끝(EOL)으로 끝나며 여기에 포함될 수 있는 4개의 절(**switch**, **case**, **default** 및 **end switch**)은 줄 경계를 넘지 않아야 합니다.*

인수:

인수

인수	설명
expression	임의의 표현식입니다.
valuelist	쉼표로 구분된 값 목록이며, 표현식의 값과 비교됩니다. 값 목록의 값이 표현식의 값과 같은 것으로 발견된 첫 번째 그룹의 문을 사용하여 스크립트 실행이 계속됩니다. 값 목록의 각 값은 임의의 표현식일 수 있습니다. **case** 절에서 일치하는 항목이 발견되지 않는 경우 **default** 절 아래의 문이 실행됩니다(지정한 경우).
statements	하나 이상의 Qlik Sense 스크립트 문의 그룹입니다.

```
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

To 스크립트 키워드는 여러 스크립트 문에서 사용됩니다.

3.2 스크립트 접두사

접두사는 해당되는 정규 문에는 적용할 수 있지만 제어 문에는 적용할 수 없습니다. 하지만 **when** 및 **unless** 접두사는 소수의 특정 제어 문 절에서 접미사로 사용할 수 있습니다.

모든 스크립트 키워드는 소문자와 대문자를 원하는 대로 조합하여 입력할 수 있습니다. 하지만 문에 사용되는 필드 및 변수 이름은 대/소문자가 구분됩니다.

스크립트 접두사 개요

각 함수는 개요가 끝난 후에 더 자세히 설명합니다. 구문에서 함수 이름을 클릭하여 해당 함수에 대한 상세 설명에 즉시 액세스할 수도 있습니다.

Add

Add 접두사를 스크립트의 **LOAD** 또는 **SELECT** 문에 추가하여 레코드를 다른 테이블에 추가하도록 지정할 수 있습니다. 또한 이 문을 부분 로드에서 실행하도록 지정합니다. **Add** 접두사는 **Map** 문에도 사용할 수 있습니다.

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

```
Add [ Only ] mapstatement
```

Buffer

QVD 파일은 **buffer** 접두사를 통해 자동으로 생성 및 유지 관리할 수 있습니다. 이 접두사는 스크립트에서 대부분의 **LOAD** 및 **SELECT** 문에 사용할 수 있습니다. 이 접두사는 해당 문의 결과를 캐시/버퍼링하는 데 QVD 파일을 사용하도록 지정합니다.

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

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

Concatenate

서로 다른 필드 집합을 가진 두 테이블을 연결하는 경우 **Concatenate** 접두사를 사용하여 두 테이블의 연결을 강제로 적용할 수 있습니다.

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

Crosstable

crosstable 로드 접두사는 "교차 표" 또는 "피벗 테이블" 구조화된 데이터를 바꾸는 데 사용됩니다. 이러한 방식으로 구조화된 데이터는 스프레드시트 소스로 작업할 때 일반적으로 발생합니다. **crosstable** 로드 접두사의 출력 및 목표는 이러한 구조를 일반 열 지향 테이블에 해당하는 것으로 바꾸는 것입니다. 이 구조는 일반적으로 Qlik Sense의 분석에 더 적합하기 때문입니다.

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

First

LOAD 또는 **SELECT (SQL)** 문의 **First** 접두사는 데이터 소스 테이블에서 최대 수의 레코드 집합을 로드하는 데 사용됩니다.

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

Generic

Generic 로드 접두사는 엔터티-특성-값 모델링 데이터(EAV)를 기존의 정규화된 관계형 테이블 구조로 변환할 수 있도록 합니다. EAV 모델링은 "일반 데이터 모델링" 또는 "개방형 스키마"라고도 합니다.

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

Hierarchy

hierarchy 접두사는 부모-자식 계층 구조 테이블을 Qlik Sense 데이터 모델에서 유용한 테이블로 변환하는 데 사용됩니다. 이 접두사는 **LOAD** 또는 **SELECT** 문 앞에 넣을 수 있으며 로드 문의 결과를 입력으로 사용하여 테이블 변환을 수행합니다.

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

HierarchBelongsTo

이 접두사는 부모-자식 계층 구조 테이블을 Qlik Sense 데이터 모델에서 유용한 테이블로 변환하는 데 사용됩니다. 이 접두사는 **LOAD** 또는 **SELECT** 문 앞에 넣을 수 있으며 로드 문의 결과를 입력으로 사용하여 테이블 변환을 수행합니다.

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

Inner

join 및 **keep** 접두사 앞에 **inner** 접두사를 사용할 수 있습니다.

join 앞에 사용할 경우 내부 조인을 사용하도록 지정됩니다. 따라서 결과 테이블은 원시 데이터 테이블의 필드 값 조합을 포함하며 연결 필드 값은 두 테이블 모두에 표시됩니다. **keep** 앞에 사용할 경우 Qlik Sense에 저장되기 전에 두 원시 데이터 테이블을 공통 교집합으로 축소하도록 지정합니다.
에서 관리합니다.

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

IntervalMatch

IntervalMatch 접두사는 불연속 숫자 값과 하나 이상의 숫자 간격을 연결하며, 선택적으로 하나 이상의 추가 키의 값을 연결하는 테이블을 만드는 데 사용됩니다.

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

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

Join

join 접두사는 로드한 테이블을 기존의 명명된 테이블이나 마지막으로 생성한 데이터 테이블과 조인합니다.

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

Keep

keep 접두사는 **join** 접두사와 비슷합니다. **join** 접두사와 마찬가지로 로드된 테이블을 기존의 명명된 테이블이나 최근에 생성한 데이터 테이블과 비교합니다. 그러나 로드된 테이블을 기존 테이블과 조인하는 것이 아니라 Qlik Sense에 저장하기 전에 테이블 데이터의 교집합을 기준으로 둘 중 한 테이블 또는 두 테이블을 모두 축소하는 효과가 있습니다. 비교는 해당하는 조인을 수행할 때와 마찬가지로 모든 공통 필드에 대한 자연 조인과 동일하게 수행됩니다. 그러나 두 테이블은 조인되지 않으며 Qlik Sense에 별도의 명명된 두 테이블로 유지됩니다.

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

Left

Join 및 **Keep** 접두사 앞에 **left** 접두사를 사용할 수 있습니다.

join 앞에 사용할 경우 왼쪽 조인을 사용하도록 지정됩니다. 결과 테이블은 원시 데이터 테이블의 필드 값 조합만을 포함하며 연결 필드 값은 첫 번째 테이블에 표시됩니다. **keep** 앞에 사용할 경우 Qlik Sense에 저장되기 전에 두 번째 원시 데이터 테이블을 첫 번째 테이블과의 공통 교집합으로 축소하도록 지정됩니다.

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

Mapping

mapping 접두사는 스크립트 실행 중 필드 값과 필드 이름을 교체하는 등의 용도로 사용할 수 있는 매핑 테이블을 만드는 데 사용됩니다.

```
매핑  ( loadstatement | selectstatement )
```

Merge

Merge 접두사를 스크립트의 **LOAD** 또는 **SELECT** 문에 추가하여 로드된 테이블을 다른 테이블에 병합하도록 지정할 수 있습니다. 또한 이 문을 부분 로드에서 실행하도록 지정합니다.

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

NoConcatenate

NoConcatenate 접두사를 사용하면 동일한 필드 집합을 포함한 두 개의 로드된 테이블이 두 개의 개별 내부 테이블로 처리됩니다. 그렇지 않은 경우 이 테이블은 자동으로 연결됩니다.

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

Outer

명시적인 **Join** 접두사 앞에 **Outer** 접두사를 사용하여 외부 조인을 지정할 수 있습니다. 외부 조인 시 두 테이블 간의 모든 조합이 생성됩니다. 따라서 결과 테이블은 원시 데이터 테이블의 필드 값 조합을 포함하며 연결 필드 값은 한 테이블이나 두 테이블 모두에 표시됩니다. **Outer** 키워드는 옵션이며 join 접두사를 지정하지 않을 경우 사용되는 기본 조인 유형입니다.

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

Partial reload
전체 로드는 항상 기존 데이터 모델의 모든 테이블을 삭제하여 시작한 다음 로드 스크립트를 실행합니다.

*부분 로드 (page 98)*는 이 작업을 수행하지 않습니다. 대신 데이터 모델의 모든 테이블을 유지한 다음 앞에 **Add**, **Merge** 또는 **Replace** 접두사가 붙은 **Load** 및 **Select** 문만 실행합니다. 다른 데이터 테이블은 명령의 영향을 받지 않습니다. **only** 인수는 문이 부분 로드 중에만 실행되어야 하며 전체 로드 중에는 무시되어야 함을 나타냅니다. 다음 테이블에는 부분 및 전체 로드에 대한 문 실행이 요약되어 있습니다.

Replace
Replace 접두사를 스크립트의 **LOAD** 또는 **SELECT** 문에 추가하여 로드된 테이블이 다른 테이블을 대체하도록 지정할 수 있습니다. 또한 이 문을 부분 로드에서 실행하도록 지정합니다. **Replace** 접두사는 **Map** 문에도 사용할 수 있습니다.

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

```
Replace [only] mapstatement
```

Right
Join 및 **Keep** 접두사 앞에 **right** 접두사를 사용할 수 있습니다.

join 앞에 사용할 경우 오른쪽 조인을 사용하도록 지정됩니다. 결과 테이블은 원시 데이터 테이블의 필드 값 조합만을 포함하며 연결 필드 값은 두 번째 테이블에 표시됩니다. **keep** 앞에 사용할 경우 Qlik Sense에 저장되기 전에 첫 번째 원시 데이터 테이블을 두 번째 테이블과의 공통 교집합으로 축소하도록 지정됩니다.

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

Sample
LOAD 또는 **SELECT** 문의 **sample** 접두사는 데이터 소스에서 무작위 레코드 샘플을 로드하는 데 사용됩니다.

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

Semantic
레코드 간의 관계가 포함된 테이블은 **semantic** 접두사를 통해 로드할 수 있습니다. 예를 들어, 테이블 내에서 자체 참조될 수 있으며, 한 레코드가 부모 등의 다른 레코드 또는 선행자를 가리키거나 속할 수 있습니다.

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

Unless
unless 접두사 및 접미사는 문 또는 exit 절을 평가할지 여부를 결정하는 조건절을 만드는 데 사용됩니다. 전체 **if..end if** 문의 간단한 대체 옵션으로 사용되는 경우도 있습니다.

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

When
when 접두사 및 접미사는 문 또는 exit 절을 실행할지 여부를 결정하는 조건절을 만드는 데 사용됩니다. 전체 **if..end if** 문의 간단한 대체 옵션으로 사용되는 경우도 있습니다.

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

Add

Add 접두사를 스크립트의 **LOAD** 또는 **SELECT** 문에 추가하여 레코드를 다른 테이블에 추가하도록 지정할 수 있습니다. 또한 이 문을 부분 로드에서 실행하도록 지정합니다. **Add** 접두사는 **Map** 문에도 사용할 수 있습니다.

부분 다시 로드가 제대로 작동하려면 부분 다시 로드가 트리거되기 전에 앱을 데이터와 함께 열어야 합니다.

로드 버튼을 사용하여 부분 로드를 수행합니다. Qlik Engine JSON API을 사용할 수도 있습니다.

구문:

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

```
Add [only] mapstatement
```

일반(부분 아님) 로드 중에는 **Add LOAD** 구조가 일반 **LOAD** 문으로 작동합니다. 레코드가 생성되어 테이블에 저장됩니다.

Concatenate 접두사가 사용되거나 동일한 필드 집합의 테이블이 있는 경우 레코드가 관련된 기존 테이블에 추가됩니다. 그렇지 않으면 **Add LOAD** 구조가 새 테이블을 만듭니다.

부분 로드도 같습니다. 유일한 차이는 **Add LOAD** 구조는 새 테이블을 만들지 않는다는 것입니다. 레코드를 추가해야 하는 이전 스크립트 실행의 관련 테이블이 항상 존재합니다.

중복 확인은 수행되지 않습니다. 따라서 **Add** 접두사를 사용하는 문에는 중복을 방지하는 distinct 한정자 또는 where 절이 포함됩니다.

Add Map...Using 문을 사용하면 부분 스크립트 실행 도중에도 매핑이 일어나게 됩니다.

인수:

인수

인수	설명
only	문이 부분 로드 중에만 실행되어야 함을 표시하는 선택적 한정자입니다. 일반(부분 아님) 로드 중에는 무시되어야 합니다.

예 및 결과:

예	결과

`Tab1:`
 	
`LOAD Name, Number FROM Persons.csv;`
 	
`Add LOAD Name, Number FROM newPersons.csv;`
	일반 재로드 중에 데이터가 *Persons.csv*에서 로드되어 Qlik Sense 테이블 Tab1에 저장됩니다. 그 다음 *NewPersons.csv*의 데이터가 동일한 Qlik Sense 테이블에 연결됩니다.
	
	부분 재로드 중에 데이터가 *NewPersons.csv*에서 로드되어 Qlik Sense 테이블 Tab1에 추가됩니다. 중복 확인은 수행되지 않습니다.

예	결과
Tab1: SQL SELECT Name, Number FROM Persons.csv; Add LOAD Name, Number FROM NewPersons.csv where not exists(Name);	이전에 로드된 테이블 데이터에 Name이 존재하는지 확인하기 위해 중복 확인이 수행됩니다. 일반 재로드 중에 데이터가 *Persons.csv*에서 로드되어 Qlik Sense 테이블 Tab1에 저장됩니다. 그 다음 *NewPersons.csv*의 데이터가 동일한 Qlik Sense 테이블에 연결됩니다. 부분 재로드 중에 데이터가 *NewPersons.csv*에서 로드되어 Qlik Sense 테이블 Tab1에 추가됩니다. 이전에 로드된 테이블 데이터에 Name이 존재하는지 확인하기 위해 중복 확인이 수행됩니다.
Tab1: LOAD Name, Number FROM Persons.csv; Add Only LOAD Name, Number FROM NewPersons.csv where not exists(Name);	일반 재로드 중에 데이터가 *Persons.csv*에서 로드되어 Qlik Sense 테이블 Tab1에 저장됩니다. *NewPersons.csv*를 로드하는 문은 무시됩니다. 부분 재로드 중에 데이터가 *NewPersons.csv*에서 로드되어 Qlik Sense 테이블 Tab1에 추가됩니다. 이전에 로드된 테이블 데이터에 Name이 존재하는지 확인하기 위해 중복 확인이 수행됩니다.

Buffer

QVD 파일은 **buffer** 접두사를 통해 자동으로 생성 및 유지 관리할 수 있습니다. 이 접두사는 스크립트에서 대부분의 **LOAD** 및 **SELECT** 문에 사용할 수 있습니다. 이 접두사는 해당 문의 결과를 캐시/버퍼링하는 데 QVD 파일을 사용하도록 지정합니다.

구문:

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

옵션이 사용되지 않으면 스크립트가 처음 실행될 때 생성된 QVD 버퍼가 무제한으로 사용됩니다.

버퍼 파일은 *Buffers* 하위 폴더(일반적으로 *C:\ProgramData\Qlik\Sense\Engine\Buffers*(서버 설치) 또는 *C:\Users\{user}\Documents\Qlik\Sense\Buffers*(Qlik Sense Desktop)에 저장됩니다.

QVD 파일의 이름은 계산된 이름으로, 다음 **LOAD** 또는 **SELECT** 문 및 기타 식별 정보 전체의 160비트 16진수 해시입니다. 따라서 QVD 버퍼는 다음 **LOAD** 또는 **SELECT** 문에서 변경이 있으면 무효가 됩니다.

일반적으로 QVD 버퍼는 앱 내에서 버퍼를 만든 전체 스크립트가 실행되는 동안 더 이상 참조되지 않거나 버퍼를 만든 앱이 더 이상 존재하지 않으면 제거됩니다.

인수:

인수

인수	설명
incremental	incremental 옵션은 원본 파일의 일부분만 읽는 기능을 사용합니다. 파일의 이전 크기는 QVD 파일의 XML 헤더에 저장됩니다. 이 기능은 특히 로그 파일에 유용합니다. 이전 항목에서 로드된 모든 레코드는 QVD 파일에서 읽어들이고 새로운 후속 레코드는 원본 소스에서 읽어들여서 최종적으로 업데이트된 QVD 파일을 만듭니다. 이 incremental 옵션은 **LOAD** 문 및 텍스트 파일에만 사용할 수 있습니다. 기존 데이터를 변경하거나 삭제하는 경우 증분 로드를 사용할 수 없습니다.
stale [after] amount [(days | hours)]	amount는 기간을 지정하는 수입니다. 소수를 사용할 수 있습니다. 단위를 생략한 경우 days가 사용됩니다. stale after 옵션은 원본 데이터에 단순 타임스탬프가 없는 DB 소스에서 일반적으로 사용됩니다. 대신 얼마나 오래된 QVD 스냅샷까지 사용 가능한지 지정합니다. stale after 절은 QVD 버퍼가 생성된 이후 기간이 얼마나 지나면 더 이상 유효하지 않은 것으로 간주할지 지정합니다. 이 기간이 되기 전까지는 QVD 버퍼가 데이터 소스로 사용되며 그 이후로는 원본 데이터 소스가 사용됩니다. 그러면 QVD 버퍼 파일이 자동으로 업데이트되고 새로운 기간이 시작됩니다.

제한 사항:

여러 제한 사항 가운데 가장 중요한 것은 모든 복합 문의 핵심에 파일 **LOAD** 또는 **SELECT** 문이 있어야 한다는 것입니다.

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

Concatenate는 데이터 집합을 이미 존재하는 인 메모리 테이블에 추가할 수 있도록 하는 스크립트 로드 접두사입니다. 여러 트랜잭션 데이터 집합을 단일 중앙 팩트 테이블에 추가하거나 여러 소스에서 비롯된 특정 유형의 공통 참조 데이터 집합을 구축하는 데 자주 사용됩니다. 기능면에서 SQL UNION 연산자와 유사합니다.

concatenate 작업의 결과 테이블에는 해당 테이블의 맨 아래에 추가된 새 데이터 행과 함께 원본 데이터 집합이 포함됩니다. 소스 및 대상 테이블에 다른 필드가 있을 수 있습니다. 필드가 다른 경우 결과 테이블은 소스 테이블과 대상 테이블에 있는 모든 필드의 결합된 결과를 나타내도록 확장됩니다.

구문:

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

인수

인수	설명
tablename	기존 테이블의 이름입니다. 명명된 테이블이 Concatenate 작업의 대상이 되며 로드된 모든 데이터 레코드가 해당 테이블에 추가됩니다. tablename 매개 변수를 사용하지 않으면 대상 테이블은 이 명령문 이전에 마지막으로 로드된 테이블이 됩니다.
loadstatement/selectstatement	tablename 인수 뒤에 오는 loadstatement/selectstatement 인수는 지정된 테이블에 연결됩니다.

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

함수 예

예	결과
Concatenate (Transactions) Load …. ;	Concatenate 접두사 아래의 LOAD 문에 로드된 데이터는 Transactions라는 기존 인메모리 테이블에 추가됩니다(로드 스크립트에서 이 지점 이전에 Transactions라는 테이블이 로드되었다고 가정).

예 1 – Concatenate 로드 접두사를 사용하여 대상 테이블에 여러 데이터 집합 추가

로드 스크립트 및 결과

개요

이 예에서는 두 개의 스크립트를 순차적으로 로드합니다.

- 첫 번째 로드 스크립트에는 Transactions라는 테이블로 전송되는 날짜와 금액이 포함된 초기 데이터 집합이 포함되어 있습니다.
- 두 번째 로드 스크립트에는 다음이 포함됩니다.

- Concatenate 접두사를 사용하여 초기 데이터 집합에 추가되는 두 번째 데이터 집합. 이 데이터 집합에는 초기 데이터 집합에 없는 추가 필드 type이 있습니다.
- Concatenate 접두사.

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

첫 번째 로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- id
- date
- amount

첫 번째 로드 스크립트 결과 테이블

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

테이블에서 초기 데이터 집합을 보여 줍니다.

두 번째 로드 스크립트

데이터 로드 편집기를 열고 아래에 로드 스크립트를 추가합니다.

```
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
];
```

결과

데이터를 로드하고 시트로 이동합니다. 이 필드를 차원으로 만듭니다.

- type

두 번째 로드 스크립트 결과 테이블

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	내부
3759	10/03/2018	384.00	외부
3760	10/06/2018	25.82	내부
3761	10/09/2018	312.00	내부
3762	10/15/2018	4.56	내부
3763	10/16/2018	90.24	내부
3764	10/18/2018	19.32	외부

type이 정의되지 않고 로드된 처음 7개 레코드에 대한 type 필드의 null 값에 유의합니다.

예 2 – 암시적 연결을 사용하여 대상 테이블에 여러 데이터 집합 추가

로드 스크립트 및 결과

개요

암시적으로 데이터를 추가하는 일반적인 사용 사례는 동일하게 구조화된 데이터의 여러 파일을 로드하고 대상 테이블에 모두 추가하려는 경우입니다.

예를 들어 다음과 같은 구문으로 파일 이름에 wildcards 사용:

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

또는 루프에서 다음과 같은 구조체 사용:

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

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

next file
```

암시적 연결은 스크립트에서 서로 정의되지 않은 경우에도 동일한 이름의 필드로 로드된 두 테이블 간에 발생합니다. 이로 인해 데이터가 의도하지 않게 테이블에 추가될 수 있습니다. 이러한 방식으로 동일한 필드가 있는 보조 테이블을 추가하지 않으려면 NoConcatenate 로드 접두사를 사용합니다. 대체 테이블 이름 태그로 테이블 이름을 바꾸는 것만으로는 암시적 연결이 발생하는 것을 방지할 수 없습니다. 자세한 내용은 NoConcatenate (page 88)을 참조하십시오.

이 예에서는 두 개의 스크립트를 순차적으로 로드합니다.

- 첫 번째 로드 스크립트에는 Transactions라는 테이블로 전송되는 4개의 필드가 있는 초기 데이터 집합이 포함되어 있습니다.
- 두 번째 로드 스크립트에는 첫 번째 데이터 집합과 동일한 필드가 있는 데이터 집합이 포함되어 있습니다.

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

첫 번째 로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- id
- date

- amount
- type

첫 번째 로드 스크립트 결과 테이블

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

테이블에서 초기 데이터 집합을 보여 줍니다.

두 번째 로드 스크립트

데이터 로드 편집기를 열고 아래에 로드 스크립트를 추가합니다.

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

결과

데이터를 로드하고 시트로 이동합니다.

두 번째 로드 스크립트 결과 테이블

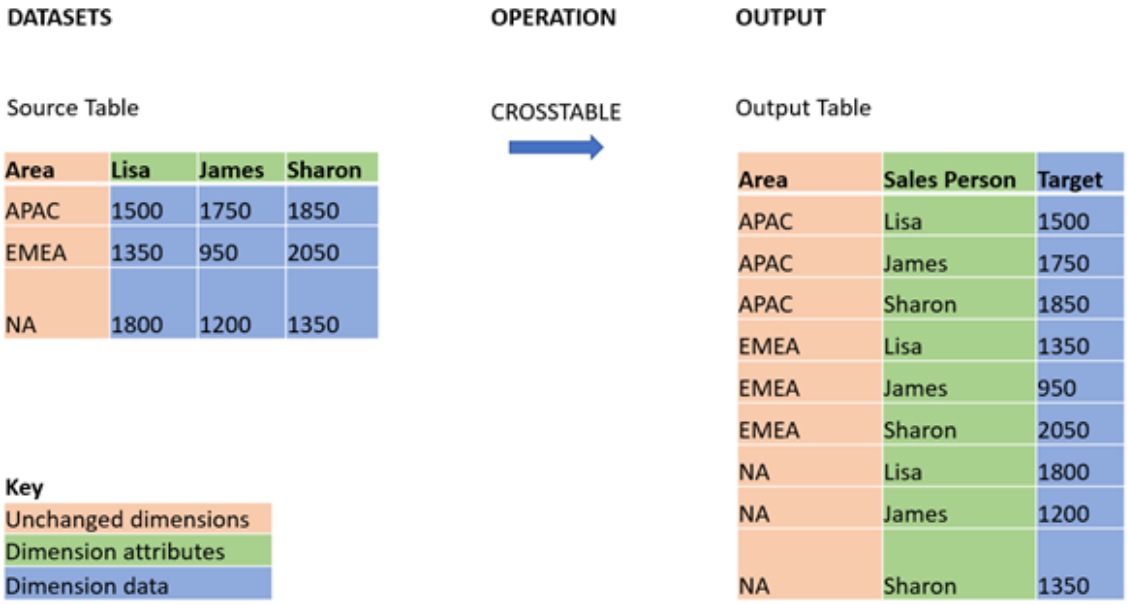
id	date	type	amount
3758	10/01/2018	내부	164.27
3759	10/03/2018	외부	384.00
3760	10/06/2018	내부	25.82
3761	10/09/2018	내부	312.00
3762	10/15/2018	내부	4.56
3763	10/16/2018	내부	90.24
3764	10/18/2018	외부	19.32
3765	11/03/2018	내부	129.40
3766	11/05/2018	외부	638.50

두 번째 데이터 집합은 동일한 필드를 가지고 있기 때문에 초기 데이터 집합에 암시적으로 연결되었습니다.

Crosstable

crosstable 로드 접두사는 "교차 표" 또는 "피벗 테이블" 구조화된 데이터를 바꾸는 데 사용됩니다. 이러한 방식으로 구조화된 데이터는 스프레드시트 소스로 작업할 때 일반적으로 발생합니다. **crosstable** 로드 접두사의 출력 및 목표는 이러한 구조를 일반 열 지향 테이블에 해당하는 것으로 바꾸는 것입니다. 이 구조는 일반적으로 Qlik Sense의 분석에 더 적합하기 때문입니다.

교차 표로 구조화된 데이터 및 교차 표 변환 후 동등한 구조의 예



구문:

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

인수

인수	설명
attribute field name	바꿀 수평 방향 차원을 설명하는 원하는 출력 필드 이름(머리글 행).
data field name	바꿀 차원의 수평 방향 데이터를 설명하는 원하는 출력 필드 이름(머리글 행 아래의 데이터 값 행렬).
n	일반 형식으로 변환할 테이블 앞에 있는 한정자 필드 또는 변경되지 않은 차원의 수. 기본값은 1입니다.

이 스크립팅 함수는 다음 함수와 관련이 있습니다.

관련 함수

함수	상호 작용
Generic (page 56)	엔터티-특성-값으로 구조화된 데이터 집합을 가져와 이를 일반 관계형 테이블 구조로 변환하는 변환 로드 접두사이며, 발생한 각 특성을 데이터의 새 필드 또는 열로 분리합니다.

예 1 – 피벗된 판매 데이터 변환(단순)

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 첫 번째 로드 스크립트를 새 탭에 추가합니다.

첫 번째 로드 스크립트에는 나중에 crosstable 스크립트 접두사가 적용될 데이터 집합이 포함되어 있으며, 이 스크립트에는 crosstable을 적용하는 섹션이 주석 처리되어 있습니다. 즉, 로드 스크립트에서 이 섹션을 비활성화하기 위해 주석 구문이 사용되었습니다.

두 번째 로드 스크립트는 첫 번째 로드 스크립트와 동일하지만 주석 처리가 제거된 crosstable의 응용 프로그램을 사용합니다(주석 구문을 제거하여 활성화됨). 이러한 스크립트는 데이터 변환에서 이 스크립팅 함수의 값을 강조하기 위해 이러한 방식으로 표시됩니다.

첫 번째 로드 스크립트(함수가 적용되지 않음)

```
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;
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- Product
- Jan 2021
- Feb 2021
- Mar 2021

- Apr 2021
- May 2021
- Jun 2021

결과 테이블

제품	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

이 스크립트를 사용하면 각 월에 대한 하나의 열과 제품당 하나의 행이 있는 교차 표를 만들 수 있습니다. 현재 형식으로는 이 데이터를 분석하기가 쉽지 않습니다. 한 필드에 모든 숫자가 있고 다른 필드(즉, 3개 열이 있는 테이블)에 모든 월이 있는 것이 훨씬 좋습니다. 다음 섹션에서는 교차 표로 이 변환을 수행하는 방법을 설명합니다.

두 번째 로드 스크립트(함수가 적용됨)

//를 제거하여 스크립트의 주석 처리를 제거합니다. 이 로드 스크립트는 다음과 같이 표시되어야 합니다.

```
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;
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- Product
- Month
- Sales

결과 테이블

제품	월	판매
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

스크립트 접두사가 적용되면 교차 표는 `Month`에 대한 열과 `Sales`에 대한 열이 별도로 있는 일반표로 변환됩니다. 이렇게 하면 데이터의 가독성이 향상됩니다.

예 2 - 피벗된 판매 목표 데이터를 수직 테이블 구조로 변환(중간)

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Targets라는 테이블에 로드되는 데이터 집합.
- 피벗된 판매 사원 이름을 `Sales Person`으로 레이블이 지정된 자체 필드로 바꾸는 `crosstable` 로드 접두사.
- `Target`이라는 필드로 구조화되는 관련 판매 목표 데이터.

로드 스크립트

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

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- Area
- Sales Person

이 측정값을 추가합니다.

```
=Sum(Target)
```

결과 테이블

영역	(판매 직원)	=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

데이터 표시를 피벗된 입력 테이블로 복제하려는 경우 시트에 동등한 피벗 테이블을 만들 수 있습니다.

다음과 같이 하십시오.

1. 방금 만든 테이블을 복사하여 시트에 붙여넣습니다.
2. 새로 만든 테이블 복사본 위로 **피벗 테이블** 차트 개체를 끕니다. **변환**을 선택합니다.
3. ✔ **편집 완료**를 클릭합니다.
4. 세로 열 선반에서 가로 열 선반으로 Sales Person 필드를 끕니다.

다음 테이블은 Qlik Sense에 표시되는 초기 테이블 형식의 데이터를 보여 줍니다.

Qlik Sense에 표시된 원본 결과 테이블

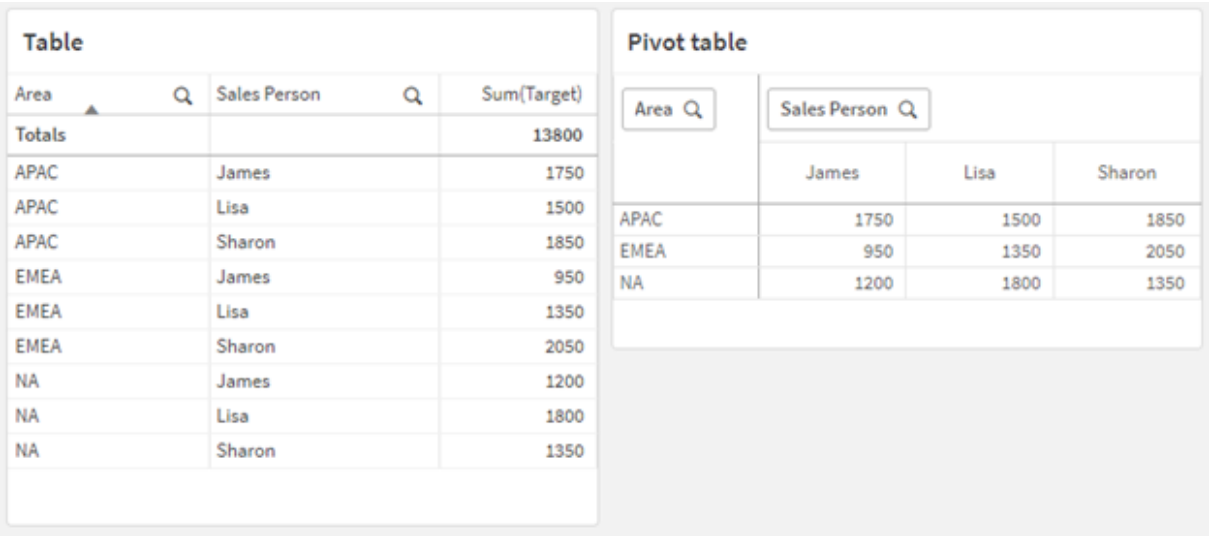
영역	(판매 직원)	=Sum(Target)
합계	-	13800
APAC	James	1750
APAC	Lisa	1500
APAC	Sharon	1850
EMEA	James	950
EMEA	Lisa	1350
EMEA	Sharon	2050
NA	James	1200
NA	Lisa	1800
NA	Sharon	1350

동등한 피벗 테이블은 다음과 유사하며, 이 테이블에는 각 판매 사원의 이름에 대한 열이 `Sales Person`에 대한 더 큰 행에 포함되어 있습니다.

`Sales Person` 필드가 가로로 피벗된 동등한 피벗 테이블

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

Sales Person 필드가 가로로 피벗된 테이블 및 동등한 피벗 테이블로 표시된 데이터의 예



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

예 3 – 피벗된 판매 및 목표 데이터를 세로 테이블 구조로 변환(고급)

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 지역 및 월별로 구성된 판매 및 목표 데이터를 나타내는 데이터 집합. 이는 SalesAndTargets라는 테이블에 로드됩니다.
- crosstable 로드 접두사. 이는 Month Year 차원을 전용 필드로 피벗 해제하고 판매 및 목표 금액 행렬을 Amount라는 전용 필드로 바꾸는 데 사용됩니다.
- 텍스트에서 날짜로의 변환 함수 date#를 사용하여 텍스트에서 적절한 날짜로 Month Year 필드 변환. 날짜로 변환된 이 Month Year 필드는 Join 로드 접두사를 통해 SalesAndTarget 테이블에 다시 조인됩니다.

로드 스크립트

```
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;
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- Area
- Month Year

레이블 Actual을 사용하여 다음 측정값을 만듭니다.

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

또한 레이블 Target을 사용하여 이 측정값을 만듭니다.

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

결과 테이블(잘림)

영역	Month Year	실제	대상
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

데이터 표시를 피벗된 입력 테이블로 복제하려는 경우 시트에 동등한 피벗 테이블을 만들 수 있습니다.

다음과 같이 하십시오.

1. 방금 만든 테이블을 복사하여 시트에 붙여넣습니다.
2. 새로 만든 테이블 복사본 위로 **피벗 테이블** 차트 개체를 끕니다. **변환**을 선택합니다.
3. ✔ **편집 완료**를 클릭합니다.
4. 세로 열 선반에서 가로 열 선반으로 `Month Year` 필드를 끕니다.
5. 가로 열 선반에서 세로 열 선반으로 `Values` 항목을 끕니다.

다음 테이블은 Qlik Sense에 표시되는 초기 테이블 형식의 데이터를 보여 줍니다.

Qlik Sense에 표시된 대로 원래 결과 테이블(잘림)

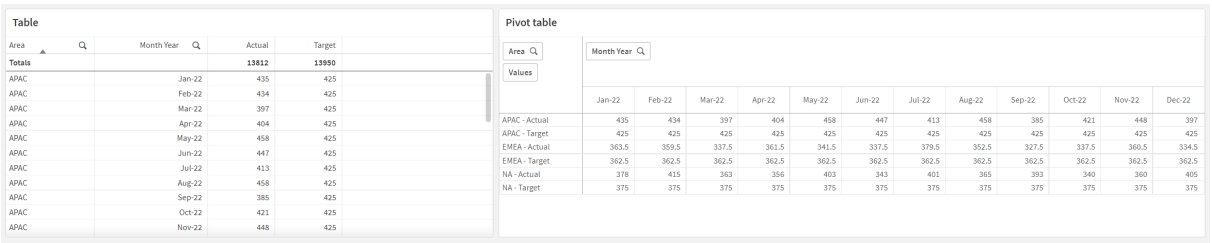
영역	Month Year	실제	대상
합계	-	13812	13950
APAC	Jan-22	435	425
APAC	Feb-22	434	425
APAC	Mar-22	397	425
APAC	Apr-22	404	425
APAC	May-22	458	425
APAC	Jun-22	447	425
APAC	Jul-22	413	425
APAC	Aug-22	458	425
APAC	Sep-22	385	425
APAC	Oct-22	421	425
APAC	Nov-22	448	425
APAC	Dec-22	397	425
EMEA	Jan-22	363.5	362.5
EMEA	Feb-22	359.5	362.5

동등한 피벗 테이블은 다음과 유사하며, 연도의 각 월에 대한 열이 `Month Year`에 대한 더 큰 행에 포함되어 있습니다.

Month Year 필드가 가로로 피벗된 동등한 피벗 테이블(잘림)

영역 (값)	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 - 실제	435	434	397	404	458	447	413	458	385	421	448	397
APAC - 대상	425	425	425	425	425	425	425	425	425	425	425	425
EMEA - 실제	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 - 대상	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 - 실제	378	415	363	356	403	343	401	365	393	340	360	405
EMEA - 대상	375	375	375	375	375	375	375	375	375	375	375	375

Month Year 필드가 가로로 피벗된 테이블 및 동등한 피벗 테이블로 표시된 데이터의 예



First

LOAD 또는 SELECT(SQL) 문의 First 접두사는 데이터 소스 테이블에서 최대 수의 레코드 집합을 로드하는 데 사용됩니다. First 접두사를 사용하는 일반적인 사용 사례는 크거나 느린 데이터 로드 단계에서 레코드의 작은 하위 집합을 검색하려는 경우에 있습니다. 정의된 "n"개의 레코드가 로드되는 즉시 로드 단계가 조기에 종료되고 나머지 스크립트 실행은 정상적으로 계속됩니다.

구문:

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

인수

인수	설명
n	읽을 최대 레코드 수를 나타내는 정수로 평가되는 임의의 표현식입니다. n은 (n)과 같이 괄호로 묶을 수도 있습니다.
loadstatement | selectstatement	n 인수 뒤에 오는 load statement/select statement는 설정된 최대 레코드 수로 로드되어야 하는 지정된 테이블을 정의합니다.

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

함수 예

예	결과
FIRST 10 LOAD * from abc.csv;	이 예는 Excel 파일에서 처음 10개의 줄을 검색합니다.
FIRST (1) SQL SELECT * from Orders;	이 예는 Orders 데이터 집합에서 첫 번째로 선택한 줄을 검색합니다.

예 – 처음 5개 행 로드

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 2020년 처음 2주 동안의 날짜 데이터 집합.
- 처음 5개 레코드만 로드하도록 응용 프로그램에 지시하는 First 변수.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고, `Date`를 필드로 추가하고, `sum(sales)`를 측정값으로 추가합니다.

결과 테이블

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

이 스크립트는 `Sales` 테이블의 처음 5개 레코드만 로드합니다.

Generic

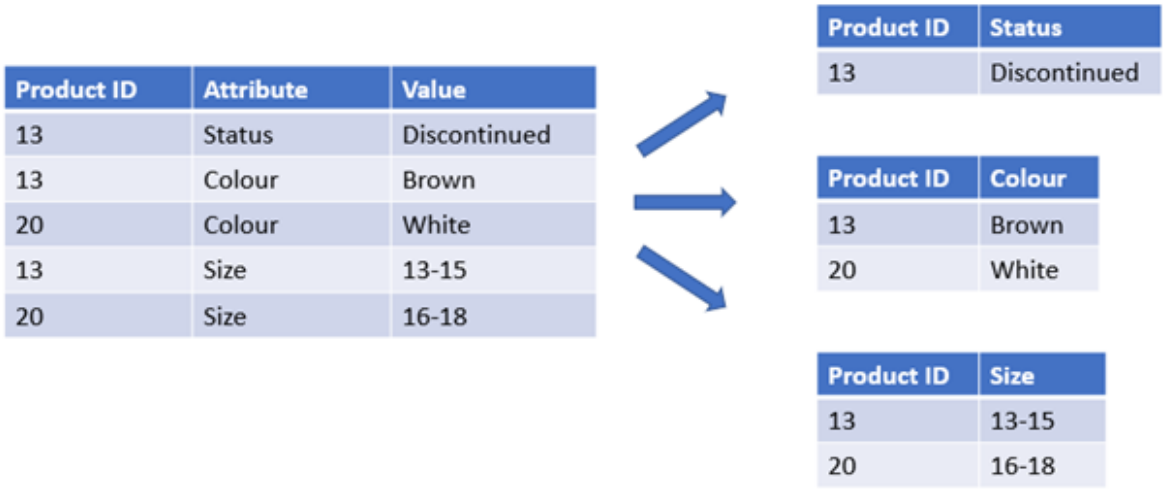
Generic 로드 접두사는 엔터티-특성-값 모델링 데이터(EAV)를 기존의 정규화된 관계형 테이블 구조로 변환할 수 있도록 합니다. EAV 모델링은 "일반 데이터 모델링" 또는 "개방형 스키마"라고도 합니다.

EAV 모델링 데이터 및 이에 상응하는 비정규화된 관계형 테이블의 예

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

EAV 모델링 데이터 및 이에 상응하는 정규화된 관계형 테이블 집합의 예



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

Qlik에서 EAV 모델링 데이터를 로드하고 분석하는 것이 기술적으로 가능하지만, 이에 상응하는 기존 관계형 데이터 구조로 작업하는 것이 더 쉬운 경우가 많습니다.

구문:

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

다음 항목은 이 함수를 사용하는 데 도움이 될 수 있습니다.

관련 항목

항목	설명
Crosstable (page 45)	Crosstable 로드 접두사는 가로 방향의 데이터를 세로 방향의 데이터로 변환합니다. 접두사는 일반적으로 완전히 다른 사용 사례를 제공하지만 ,순전히 기능적 관점에서 Generic 로드 접두사에 대해 반대 변환을 수행합니다.
*데이터 관리*의 **일반 데이터베이스**	EAV로 구조화된 데이터 모델은 여기에서 자세히 설명합니다.

예 1 - 일반 로드 접두사를 사용하여 EAV로 구조화된 데이터 변환

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

이 로드 스크립트에는 Transactions라는 테이블에 로드되는 데이터 집합이 포함되어 있습니다. 이 데이터 집합에는 날짜 필드가 포함됩니다. 기본 MonthNames 정의가 사용됩니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다. `Color` .

이 측정값을 추가합니다.

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

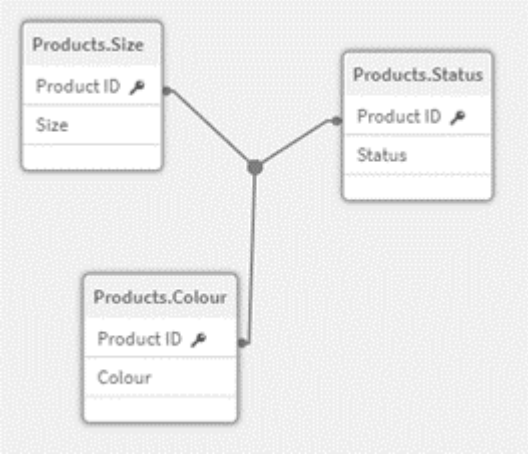
이제 색 기준으로 제품 수를 검사할 수 있습니다.

결과 테이블

색	**=Count([제품 ID])**
갈색	4
흰색	2

각 특성이 원본 대상 테이블 태그 `Product`에 따라 명명된 별도의 테이블로 분할된 데이터 모델의 모양에 유의합니다. 각 테이블에는 특성이 접미사로 있습니다. 예를 들어 `Product.Color`입니다. 결과로 제공되는 제품 특성 출력 레코드는 `Product ID`로 연결됩니다.

결과의 데이터 모델 뷰어 표현



결과 레코드 테이블: 제품 상태

제품 ID	상태
13	중단됨
2	중단됨

결과 레코드 테이블: 제품.크기

제품 ID	크기
13	13-15
20	16-18
45	16-18

결과 레코드 테이블: 제품 색

제품 ID	색
13	갈색
5	갈색
44	갈색
45	갈색
20	흰색
2	흰색

예 2 - 일반 로드 접두사 없이 EAV로 구조화된 데이터 분석

로드 스크립트 및 차트 표현식

개요

이 예는 EAV로 구조화된 데이터를 원본 형태로 분석하는 방법을 보여 줍니다.

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

이 로드 스크립트에는 EAV 구조의 Products라는 테이블에 로드되는 데이터 집합이 포함되어 있습니다.

이 예에서는 여전히 색상 특성별로 제품을 계수하고 있습니다. 이러한 방식으로 구조화된 데이터를 분석하려면 특성 값 Color를 전달하는 제품의 표현식 수준 필터링을 적용해야 합니다.

또한 개별 특성을 차원이나 필드로 선택할 수 없으므로 효과적인 시각화를 구축하는 방법을 결정하기가 더 어렵습니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다. Value .

다음 측정값을 만듭니다.

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

이제 색 기준으로 제품 수를 검사할 수 있습니다.

결과 레코드 테이블: 제품 상태

값	=Count({<Attribute={'Color'}>} [제품 ID])
갈색	4
흰색	2

예 3 – 일반 로드에서 결과 출력 테이블 비정규화(고급)

로드 스크립트 및 차트 표현식

개요

이 예에서는 Generic 로드 접두사에 의해 생성된 정규화된 데이터 구조가 통합된 Product 차원 테이블로 다시 비정규화될 수 있는 방법을 보여 줍니다. 이는 데이터 모델 성능 조정의 일부로 사용할 수 있는 고급 모델링 기술입니다.

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트

```
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;
```

결과

데이터 모델 뷰어를 열고 결과 데이터 모델의 모양을 확인합니다. 하나의 비정규화된 테이블만 있습니다. 이는 세 개의 중간 출력 테이블(Products.Size, Products.Status 및 Products.Color)의 조합입니다.

결과 내부 데이터 모델

제품
제품 ID
상태
색
크기

결과 레코드 테이블: 제품

제품 ID	상태	색	크기
13	중단됨	갈색	13-15
20	-	흰색	16-18
2	중단됨	흰색	-
5	-	갈색	-
44	-	갈색	-
45	-	갈색	16-18

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다. `Color`.

이 측정값을 추가합니다.

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

결과 테이블

색	=Count([제품 ID])
갈색	4
흰색	2

Hierarchy

hierarchy 접두사는 부모-자식 계층 구조 테이블을 Qlik Sense 데이터 모델에서 유용한 테이블로 변환하는 데 사용됩니다. 이 접두사는 **LOAD** 또는 **SELECT** 문 앞에 넣을 수 있으며 로드 문의 결과를 입력으로 사용하여 테이블 변환을 수행합니다.

이 접두사를 사용하면 일반적으로 입력 테이블과 같은 수의 레코드를 포함하지만 추가로 계층 구조의 각 수준을 별도의 필드에 저장하는 확장 노드 테이블이 생성됩니다. 경로 필드는 트리 구조에서 사용할 수 있습니다.

구문:

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

입력 테이블은 인접 노드 테이블이어야 합니다. 인접 노드 테이블이란 각 레코드가 한 노드에 해당하고 부모 노드에 대한 참조를 포함하는 필드를 가지는 테이블입니다. 이러한 테이블에서 노드는 한 레코드에만 저장되지만 노드가 가질 수 있는 자식의 수에는 제한이 없습니다. 물론 테이블은 노드의 특성을 나타내는 추가 필드를 포함할 수 있습니다.

이 접두사를 사용하면 일반적으로 입력 테이블과 같은 수의 레코드를 포함하지만 추가로 계층 구조의 각 수준을 별도의 필드에 저장하는 확장 노드 테이블이 생성됩니다. 경로 필드는 트리 구조에서 사용할 수 있습니다.

일반적으로 입력 테이블은 정확하게 노드당 한 레코드를 가지며 이 경우 출력 테이블도 동일한 수의 레코드를 포함합니다. 그러나 입력 테이블에서 한 노드가 여러 레코드로 표시되는 경우, 즉 여러 부모를 가진 노드가 있을 수 있습니다. 이 경우 출력 테이블의 레코드 수가 입력 테이블보다 많을 수 있습니다.

노드 ID 열에 부모 ID가 없는 모든 노드(부모 ID가 누락된 노드 포함)는 루트로 간주됩니다. 또한 루트 노드와 직간접적으로 연결된 노드만 로드되므로 순환 참조를 피할 수 있습니다.

부모 노드의 이름, 노드의 경로, 그리고 노드의 깊이를 포함하는 추가 필드를 만들 수 있습니다.

인수:

인수

인수	설명
NodeID	노드 ID가 포함된 필드의 이름입니다. 이 필드는 입력 테이블에 존재해야 합니다.
ParentID	부모 노드의 노드 ID가 포함된 필드의 이름입니다. 이 필드는 입력 테이블에 존재해야 합니다.
NodeName	노드의 이름이 포함된 필드의 이름입니다. 이 필드는 입력 테이블에 존재해야 합니다.
ParentName	새 **ParentName** 필드의 이름을 지정하기 위해 사용되는 문자열입니다. 생략된 경우 이 필드가 생성되지 않습니다.
ParentSource	노드 경로를 만드는 데 사용되는 노드의 이름이 포함된 필드의 이름입니다. 선택적인 매개 변수입니다. 생략된 경우 **NodeName**이 사용됩니다.
PathName	새 **Path** 필드의 이름을 지정하는 데 사용되는 문자열로, 루트부터 해당 노드까지의 경로가 포함됩니다. 선택적인 매개 변수입니다. 생략된 경우 이 필드가 생성되지 않습니다.
PathDelimiter	새 **Path** 필드에서 구분 기호로 사용되는 문자열입니다. 선택적인 매개 변수입니다. 생략된 경우 '/'가 사용됩니다.
Depth	새 **Depth** 필드의 이름을 지정하는 데 사용되는 문자열로, 계층 구조에서 노드의 깊이가 포함됩니다. 선택적인 매개 변수입니다. 생략된 경우 이 필드가 생성되지 않습니다.

```
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

이 접두사는 부모-자식 계층 구조 테이블을 Qlik Sense 데이터 모델에서 유용한 테이블로 변환하는 데 사용됩니다. 이 접두사는 **LOAD** 또는 **SELECT** 문 앞에 넣을 수 있으며 로드 문의 결과를 입력으로 사용하여 테이블 변환을 수행합니다.

이 접두사를 사용하면 계층 구조의 모든 상위-자식 관계를 포함하는 테이블이 생성됩니다. 그러면 상위 필드를 사용하여 계층 구조의 전체 트리를 선택할 수 있습니다. 대부분의 경우 출력 테이블은 노드당 여러 레코드를 포함합니다.

구문:

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

입력 테이블은 인접 노드 테이블이어야 합니다. 인접 노드 테이블이란 각 레코드가 한 노드에 해당하고 부모 노드에 대한 참조를 포함하는 필드를 가지는 테이블입니다. 이러한 테이블에서 노드는 한 레코드에만 저장되지만 노드가 가질 수 있는 자식의 수에는 제한이 없습니다. 물론 테이블은 노드의 특성을 나타내는 추가 필드를 포함할 수 있습니다.

이 접두사를 사용하면 계층 구조의 모든 상위-자식 관계를 포함하는 테이블이 생성됩니다. 그러면 상위 필드를 사용하여 계층 구조의 전체 트리를 선택할 수 있습니다. 대부분의 경우 출력 테이블은 노드당 여러 레코드를 포함합니다.

노드의 깊이 차이를 포함하는 추가 필드를 만들 수 있습니다.

인수:

인수

인수	설명
NodeID	노드 ID가 포함된 필드의 이름입니다. 이 필드는 입력 테이블에 존재해야 합니다.
ParentID	부모 노드의 노드 ID가 포함된 필드의 이름입니다. 이 필드는 입력 테이블에 존재해야 합니다.
NodeName	노드의 이름이 포함된 필드의 이름입니다. 이 필드는 입력 테이블에 존재해야 합니다.
AncestorID	상위 노드의 ID를 포함하는 새 상위 ID 필드의 이름을 지정하는 데 사용되는 문자열입니다.
AncestorName	상위 노드의 이름을 포함하는 새 상위 필드의 이름을 지정하는 데 사용되는 문자열입니다.
DepthDiff	계층 구조에서 상위 노드에 상대적인 노드의 깊이를 포함하는 새 **DepthDiff** 필드의 이름을 지정하는 데 사용되는 문자열입니다. 선택적인 매개 변수입니다. 생략된 경우 이 필드가 생성되지 않습니다.

```
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

join 및 **keep** 접두사 앞에 **inner** 접두사를 사용할 수 있습니다. **join** 앞에 사용할 경우 내부 조인을 사용하도록 지정됩니다. 따라서 결과 테이블은 원시 데이터 테이블의 필드 값 조합을 포함하며 연결 필드 값은 두 테이블 모두에 표시됩니다. **keep** 앞에 사용할 경우 Qlik Sense에 저장되기 전에 두 원시 데이터 테이블을 공통 교집합으로 축소하도록 지정합니다.

구문:

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

인수:

인수

인수	설명
tablename	로드된 테이블과 비교할 명명된 테이블입니다.
loadstatement또는 selectstatement	로드된 테이블에 사용되는 **LOAD** 또는 **SELECT** 문입니다.

예

로드 스크립트

예제 스크립트를 앱에 추가하고 실행합니다. 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.

```
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 ];
```

결과

결과 테이블

Column1	Column2	Column3
A	B	C
1	aa	xx

설명

이 예에서는 첫 번째(왼쪽) 테이블과 두 번째(오른쪽) 테이블에 있는 값만 조인되는 내부 조인 출력을 보여 줍니다.

IntervalMatch

IntervalMatch 접두사는 불연속 숫자 값과 하나 이상의 숫자 간격을 연결하며, 선택적으로 하나 이상의 추가 키의 값을 연결하는 테이블을 만드는 데 사용됩니다.

구문:

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

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

IntervalMatch 접두사는 간격을 로드하는 **LOAD** 또는 **SELECT** 문 앞에 와야 합니다. **IntervalMatch** 접두사가 있는 문을 실행하기 전에 불연속 데이터 포인트(아래 예에서는 Time)를 포함하는 필드 및 추가 키를 Qlik Sense로 로드해야 합니다. 이 접두사가 자체적으로 필드를 데이터베이스 테이블에서 읽는 것은 아닙니다. 이 접두사는 로드한 간격과 키의 테이블을 불연속 숫자 데이터 포인트의 추가 열을 포함하는 테이블로 변환합니다. 또한 테이블이 불연속 데이터 포인트, 간격 및 키 필드의 값에 대한 가능한 조합마다 하나의 레코드를 가지도록 레코드의 수를 확장합니다.

간격은 겹칠 수 있으며 불연속 값은 일치하는 모든 간격에 연결됩니다.

IntervalMatch 접두사가 키 필드를 사용하여 확장된 경우, 하나 이상의 불연속 숫자 값과 하나 이상의 숫자 간격을 연결하는 동시에 하나 이상의 추가 키의 값을 연결하는 테이블을 만드는 데 사용됩니다.

정의되지 않은 간격 한계가 무시되지 않도록 하려면 NULL 값을 간격의 하한이나 상한이 되는 다른 필드에 매핑해야 합니다. 이를 위해서는 **NullAsValue** 문을 사용하거나 불연속 숫자 데이터 포인트를 사용하기 전후에 NULL 값을 숫자 값으로 대체하는 명시적인 테스트를 수행할 수 있습니다.

인수:

인수

인수	설명
matchfield	간격과 연결할 불연속 숫자 값을 포함하는 필드입니다.
keyfield	변환 시 일치시킬 추가 특성이 포함된 필드입니다.

인수	설명
loadstatement orselectstatement	결과는 테이블이어야 합니다. 여기서, 첫 번째 필드는 각 간격의 하한 값이 포함되고 두 번째 필드에는 각 간격의 상한 값이 포함됩니다. 또한 일치 키를 사용하는 경우 세 번째 및 후속 필드에는 **IntervalMatch** 문으로 제공된 키 필드가 포함됩니다. 간격은 항상 완료된 상태입니다. 즉, 간격에는 종료 지점이 포함되어야 합니다. 숫자가 아닌 한계를 지정하면 간격이 무시됩니다(정의되지 않음).

Example 1:

아래의 두 테이블 중 첫 번째 테이블은 여러 불연속 이벤트를 나열하고 두 번째 테이블은 다른 주문의 생산 시작 및 종료 시간을 정의합니다. **IntervalMatch** 접두사를 사용하면 두 테이블을 논리적으로 연결하여, 예를 들어 생산 중단으로 영향을 받은 주문이나 특정 주문 제품을 생산한 근무조를 확인할 수 있습니다.

```
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;
```

이제 **OrderLog** 테이블에는 추가적인 열인 *Time*이 포함됩니다. 레코드 수 역시 확장됩니다.

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

Time	Start	End	Order
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: (keyfield 사용)

위와 동일한 예로, *ProductionLine* 을 키 필드로 추가합니다.

```
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;
```

이제 아래와 같은 테이블 상자가 생성됩니다.

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

join 접두사는 로드한 테이블을 기존의 명명된 테이블이나 마지막으로 생성한 데이터 테이블과 조인합니다.

데이터를 결합하면 필드 또는 특성의 추가 집합, 즉 대상 테이블에 아직 존재하지 않는 항목으로 대상 테이블을 확장하는 효과가 있습니다. 소스 데이터 집합과 대상 테이블 사이의 모든 공통 필드 이름은 새 수신 레코드를 연결하는 방법을 해결하는 데 사용됩니다. 이를 일반적으로 "자연 조인"이라고 합니다. Qlik 조인 작업으로 인해 조인 연결의 고유성과 사용된 조인 유형에 따라 처음보다 많거나 적은 레코드가 있는 결과 대상 테이블이 생성될 수 있습니다.

네 가지 유형의 조인이 있습니다.

왼쪽 조인

왼쪽 조인은 가장 일반적인 조인 유형입니다. 예를 들어 트랜잭션 데이터 집합이 있고 이를 참조 데이터 집합과 결합하려는 경우 일반적으로 `Left Join`을 사용합니다. 먼저 트랜잭션 테이블을 로드한 다음 `Left Join` 접두사를 통해 조인하는 동안 참조 데이터 집합을 이미 로드된 트랜잭션 테이블에 로드합니다. `Left Join`은 모든 트랜잭션을 있는 그대로 유지하고 일치하는 항목이 있는 보충 참조 데이터 필드에 추가합니다.

내부 조인

일치하는 연관이 있는 결과에만 관심이 있는 두 개의 데이터 집합이 있는 경우 `Inner Join`을 사용하는 것이 좋습니다. 이렇게 하면 일치하는 항목이 없으면 로드된 소스 데이터와 대상 테이블에서 모든 레코드가 제거됩니다. 결과적으로 조인 작업이 발생하기 전보다 적은 수의 레코드가 대상 테이블에 남을 수 있습니다.

외부 조인

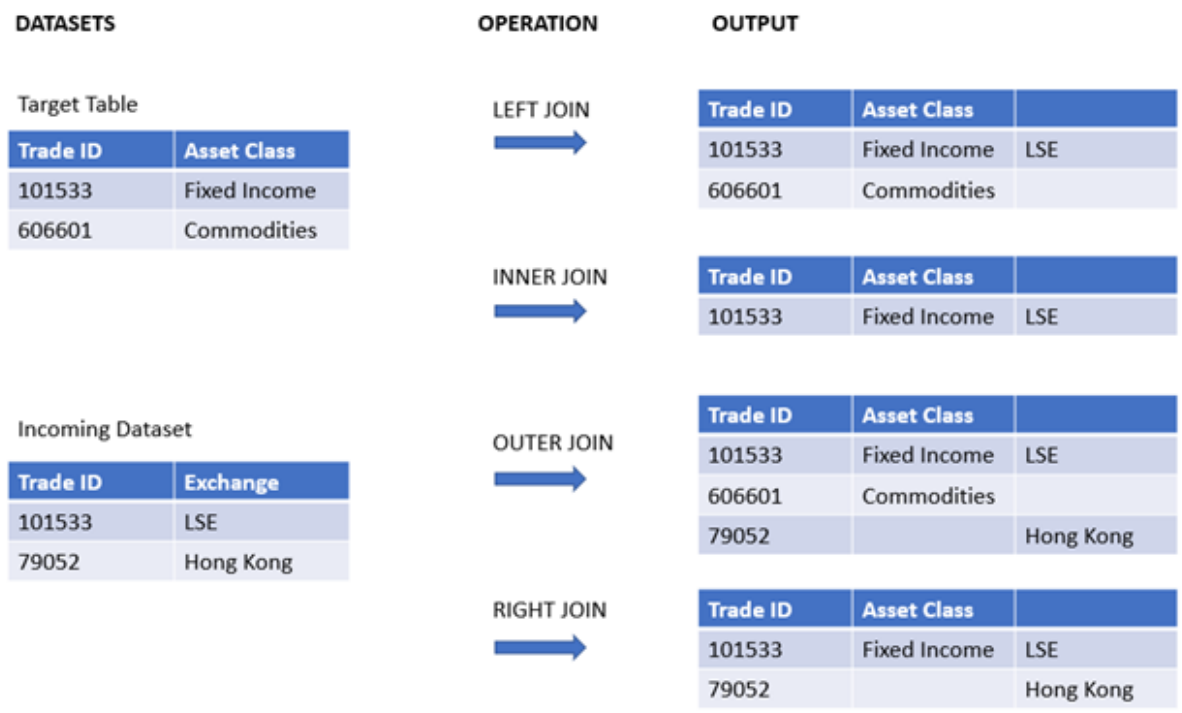
대상 레코드와 모든 수신 레코드를 모두 유지해야 하는 경우 `Outer Join`을 사용합니다. 일치하는 항목이 없는 경우 각 레코드 집합은 계속 유지되는 반면 조인 반대쪽의 필드는 채워지지 않은 상태(null)로 유지됩니다.

유형 키워드를 생략하면 기본 조인 유형은 외부 조인입니다.

오른쪽 조인

이 조인 유형은 로드될 모든 레코드를 유지하면서 조인의 대상이 되는 테이블의 레코드를 수신 레코드에 연관 일치가 있는 레코드로만 축소합니다. 이는 때때로 필요한 하위 집합에 이미 미리 로드된 레코드 테이블을 자르는 수단으로 사용하는 틈새 조인 유형입니다.

다양한 유형의 조인 작업에서 얻은 결과 집합 예



조인 작업의 소스와 대상 간에 공통 필드 이름이 없는 경우 조인은 모든 행의 데카르트 곱을 생성합니다. 이를 "교차 조인"이라고 합니다.

"교차 조인" 작업의 결과 집합 예

구문:

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

인수

인수	설명
tablename	로드된 테이블과 비교할 명명된 테이블입니다.
loadstatement또는 selectstatement	로드된 테이블에 사용되는 **LOAD** 또는 **SELECT** 문입니다.

다음 항목은 이 함수를 사용하는 데 도움이 될 수 있습니다.

관련 항목

항목	설명
*데이터 관리*에서 **조인 및 유지로 테이블 결합**	이 항목에서는 "조인" 및 "유지"하는 데이터 집합의 개념에 대한 추가 설명을 제공합니다.
Keep (page 80)	Keep 로드 접두사는 Join 접두사와 유사하지만 소스 및 대상 데이터 집합을 결합하지 않습니다. 대신 채택된 작업 유형(내부, 외부, 왼쪽 또는 오른쪽)에 따라 각 데이터 집합을 자릅니다.

예 1 - 왼쪽 조인: 참조 데이터 집합으로 대상 테이블 강화

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Changes라는 테이블에 로드되는 변경 기록을 나타내는 데이터 집합. 여기에는 Status ID 키 필드가 포함됩니다.
- 변경 상태를 나타내는 두 번째 데이터 집합. 왼쪽 Join 로드 접두사와 조인하여 로드되고 원본 변경 레코드와 결합됩니다.

이 왼쪽 조인은 공통 Status ID를 기반으로 수신 상태 레코드의 일치 항목이 발견되는 상태 특성을 추가하는 동안 변경 레코드가 그대로 유지되도록 합니다.

로드 스크립트

```
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');
```

결과

데이터 모델 뷰어를 열고 데이터 모델의 모양을 확인합니다. 하나의 비정규화된 테이블만 있습니다. 이는 각 변경 레코드에 조인된 일치하는 상태 특성과 모든 원본 변경 레코드의 조합입니다.

결과 내부 데이터 모델

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

데이터 모델 뷰어에서 미리 보기 창을 확장하면 이 전체 결과 집합의 일부가 테이블로 구성된 것을 볼 수 있습니다.

데이터 모델 뷰어에서 Changes 테이블 미리 보기

Change ID	**Status ID**	**Scheduled Start Date**	**Scheduled End Date**	**Business Impact**	**Status**	**Sub Status**
10030	4	19/01/2022	23/02/2022	None	Closed	Cancelled

Change ID	Status ID	Scheduled Start Date	Scheduled End Date	Business Impact	Status	Sub Status
10031	3	20/01/2022	25/03/2022	Low	Closed	완료됨
10015	3	04/01/2022	15/02/2022	Low	Closed	완료됨
10103	1	02/04/2022	29/05/2022	Medium	Open	Not Started
10116	1	15/04/2022	24/04/2022	None	Open	Not Started
10134	1	03/05/2022	08/07/2022	Low	Open	Not Started
10264	1	10/09/2022	17/10/2022	Medium	Open	Not Started
10040	1	29/01/2022	22/04/2022	None	Open	Not Started
10323	1	08/11/2022	26/11/2022	High	Open	Not Started
10187	2	25/06/2022	24/08/2022	Low	Open	Started
10185	2	23/06/2022	08/09/2022	None	Open	Started
10220	2	28/07/2022	06/09/2022	None	Open	Started
10326	2	11/11/2022	05/12/2022	None	Open	Started
10138	2	07/05/2022	03/08/2022	None	Open	Started
10334	2	19/11/2022	06/02/2023	Low	Open	Started

Status 테이블의 다섯 번째 행(Status ID: '5', Status: 'Closed', Sub Status: 'Obsolete')이 Changes 테이블의 레코드와 일치하지 않으므로 이 행의 정보는 위 결과 집합에 표시되지 않습니다.

데이터 로드 편집기로 돌아갑니다. 데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다. `Status`.

이 측정값을 추가합니다.

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

이제 상태에 따른 변경 수를 검사할 수 있습니다.

결과 테이블

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

예 2 – 내부 조인: 일치하는 레코드만 결합

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Changes라는 테이블에 로드되는 변경 기록을 나타내는 데이터 집합.
- 소스 시스템 JIRA에서 발생한 변경 레코드를 나타내는 두 번째 데이터 집합. Inner Join 로드 접두사와 조인하여 로드되고 원본 레코드와 결합됩니다.

이 Inner Join을 통해 두 데이터 집합에서 모두 5개의 변경 레코드만 유지됩니다.

로드 스크립트

```
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');
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- Source System
- Change ID
- Business Impact

이제 5개의 결과 레코드를 검사할 수 있습니다. Inner Join의 결과 테이블에는 두 데이터 집합의 정보가 일치하는 레코드만 포함됩니다.

결과 테이블

Source System	Change ID	Business Impact
JIRA	10030	None
JIRA	10134	Low
JIRA	10220	None
JIRA	10323	High
JIRA	10334	Low

예 3 – 외부 조인: 겹치는 레코드 집합 결합

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Changes라는 테이블에 로드되는 변경 기록을 나타내는 데이터 집합.
- 소스 시스템 JIRA에서 발생하는 변경 레코드를 나타내는 두 번째 데이터 집합. Outer Join 로드 접두사와 조인하여 로드되고 원본 레코드와 결합됩니다.

이렇게 하면 두 데이터 집합에서 겹치는 모든 변경 레코드가 유지됩니다.

로드 스크립트

```
// 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');
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- Source System
- Change ID
- Business Impact

이제 10개의 결과 레코드를 검사할 수 있습니다.

결과 테이블

Source System	Change ID	Business Impact
JIRA	10030	None
JIRA	10134	Low
JIRA	10220	None
JIRA	10323	-
JIRA	10334	Low
JIRA	10597	-

Source System	Change ID	Business Impact
-	10015	Low
-	10031	Low
-	10040	None
-	10138	None

예 4 – 오른쪽 조인: 보조 마스터 데이터 집합으로 대상 테이블 자르기

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Changes라는 테이블에 로드되는 변경 기록을 나타내는 데이터 집합.
- 소스 시스템 Teamwork에서 발생한 변경 레코드를 나타내는 두 번째 데이터 집합입니다. 이는 로드되어 Right Join 로드 접두사와 조인하여 원본 레코드와 조인됩니다.

이렇게 하면 Teamwork 변경 레코드만 유지되지며, 대상 테이블에 일치하는 Change ID가 없는 경우 Teamwork 레코드는 손실되지 않습니다.

로드 스크립트

```
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');
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- Source System
- Change ID
- Business Impact

이제 5개의 결과 레코드를 검사할 수 있습니다.

결과 테이블

Source System	Change ID	Business Impact
Teamwork	10015	Low
Teamwork	10031	Low
Teamwork	10040	None
Teamwork	10103	Medium
Teamwork	50231	-

Keep

keep 접두사는 **join** 접두사와 비슷합니다. **join** 접두사와 마찬가지로 로드된 테이블을 기존의 명명된 테이블이나 최근에 생성한 데이터 테이블과 비교합니다. 그러나 로드된 테이블을 기존 테이블과 조인하는 것이 아니라 Qlik Sense에 저장하기 전에 테이블 데이터의 교집합을 기준으로 둘 중 한 테이블 또는 두 테이블을 모두 축소하는 효과가 있습니다. 비교는 해당하는 조인을 수행할 때와 마찬가지로 모든 공통 필드에 대한 자연 조인과 동일하게 수행됩니다. 그러나 두 테이블은 조인되지 않으며 Qlik Sense에 별도의 명명된 두 테이블로 유지됩니다.

구문:

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

keep 접두사 앞에는 **inner**, **left** 또는 **right** 접두사 중 하나가 와야 합니다.

Qlik Sense 스크립트 언어에서 명시적인 **join** 접두사는 두 테이블의 완전 조인을 수행합니다. 결과는 한 테이블입니다. 이러한 조인의 결과로 매우 큰 테이블이 만들어지는 경우가 많습니다. Qlik Sense의 주요 기능 중에는 여러 테이블을 조인하는 대신 테이블을 연결하여 메모리 사용량 대폭 축소, 처리 속도 향상, 그리고 엄청난 유연성 향상을 제공하는 기능이 있습니다. 따라서 명시적 조인은 Qlik Sense 스크립트에서 일반적으로 피하는 것이 바람직합니다. keep 기능은 명시적 조인을 사용해야 하는 경우의 수를 줄이기 위해 설계되었습니다.

인수:

인수

인수	설명
tablename	로드된 테이블과 비교할 명명된 테이블입니다.
loadstatement또는 selectstatement	로드된 테이블에 사용되는 **LOAD** 또는 **SELECT** 문입니다.

```
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

Join 및 **Keep** 접두사 앞에 **left** 접두사를 사용할 수 있습니다.

join 앞에 사용할 경우 왼쪽 조인을 사용하도록 지정됩니다. 결과 테이블은 원시 데이터 테이블의 필드 값 조합만을 포함하며 연결 필드 값은 첫 번째 테이블에 표시됩니다. **keep** 앞에 사용할 경우 Qlik Sense에 저장되기 전에 두 번째 원시 데이터 테이블을 첫 번째 테이블과의 공통 교집합으로 축소하도록 지정됩니다.

같은 이름의 문자열 함수를 찾고 계십니까? 참조: Left (page 1405)

구문:

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

인수:

인수

인수	설명
tablename	로드된 테이블과 비교할 명명된 테이블입니다.
loadstatement또는 selectstatement	로드된 테이블에 사용되는 **LOAD** 또는 **SELECT** 문입니다.

예

로드 스크립트

예제 스크립트를 앱에 추가하고 실행합니다. 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.

```
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 ];
```

결과

결과 테이블

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

설명

이 예에서는 첫 번째(왼쪽) 테이블에 있는 값만 조인되는 왼쪽 조인 출력을 보여 줍니다.

매핑

mapping 접두사는 스크립트 실행 중 필드 값과 필드 이름을 교체하는 등의 용도로 사용할 수 있는 매핑 테이블을 만드는 데 사용됩니다.

구문:

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

mapping 접두사는 **LOAD** 또는 **SELECT** 문 앞에 넣을 수 있으며 로드 문의 결과를 매핑 테이블로 저장합니다. 매핑은 스크립트 실행 중에 US, U.S. 또는 America를 USA로 바꾸는 등, 필드 값을 효율적으로 대체하는 방법을 제공합니다. 매핑 테이블은 두 개의 열, 즉 비교 값을 포함하는 첫째 열과 원하는 매핑 값을 포함하는 둘째 열로 구성됩니다. 매핑 테이블은 메모리에 임시로 저장되며 스크립트 실행 후 자동으로 삭제됩니다.

매핑 테이블의 내용은 **Map ... Using** 문, **Rename Field** 문, **Applymap()** 함수 또는 **Mapsubstring()** 함수 등을 사용하여 액세스할 수 있습니다.

이 예에서는 거주 국가를 나타내는 국가 코드에 따라 영업직원 목록을 로드합니다. 국가 코드와 국가를 매핑하는 테이블을 사용하여 국가 코드를 국가명으로 바꿉니다. 매핑 테이블에는 3개 국가만 정의되어 있으며 다른 국가 코드는 'Rest of the world'로 매핑됩니다.

```
// 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';
```

결과 테이블은 다음과 같습니다.

Mapping table

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

Merge

Merge 접두사를 스크립트의 **LOAD** 또는 **SELECT** 문에 추가하여 로드된 테이블을 다른 테이블에 병합하도록 지정할 수 있습니다. 또한 이 문을 부분 로드에서 실행하도록 지정합니다.

일반적인 사용 사례로 변경 로그를 로드하고 이를 사용하여 기존 테이블에 inserts, updates 및 deletes를 적용합니다.

> *부분 다시 로드가 제대로 작동하려면 부분 다시 로드가 트리거되기 전에 앱을 데이터와 함께 열어야 합니다.*

로드 버튼을 사용하여 부분 로드를 수행합니다. Qlik Engine JSON API을 사용할 수도 있습니다.

구문:

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

인수:

인수

인수	설명
only	문이 부분 로드 중에만 실행되어야 함을 표시하는 선택적 한정자입니다. 일반(부분 아님) 로드 중에는 문이 무시됩니다.
SequenceNoField	작업 순서를 정의하는 타임스탬프 또는 시퀀스 번호가 포함된 필드의 이름입니다.
SequenceNoVar	병합되는 테이블의 SequenceNoField의 최댓값이 할당되는 변수의 이름입니다.
ListOfKeys	기본 키를 지정하는 쉼표로 구분된 필드 이름 목록입니다.
Operation	LOAD 문의 첫 번째 필드에는 'Insert', 'Update' 또는 'Delete'와 같은 텍스트 문자열 작업이 있어야 합니다. ‘i’, ‘u’ 및 ‘d’도 허용됩니다.

일반 기능

일반(부분 아님) 로드 중에 **Merge LOAD** 구조는 일반 **Load** 문으로 작동하지만 오래된 레코드 및 삭제 표시된 레코드를 제거하는 추가 기능으로 작동합니다. **Load** 문의 첫 번째 필드에는 작업에 대한 정보가 있어야 합니다. Insert, Update 또는 Delete

로드된 각 레코드에서 레코드 식별자가 이전에 로드된 레코드와 비교되고 최신 레코드(순서 번호에 따라)만 유지됩니다. 최신 레코드가 Delete로 표시되면 유지되는 레코드가 없습니다.

대상 테이블

필드 집합으로 결정되는 수정할 테이블입니다. 동일한 필드 집합이 있는 테이블(첫 번째 필드인 작업 필드 제외)이 이미 존재하는 경우 이 테이블이 수정할 관련 테이블이 됩니다. 또는 **Concatenate** 접두사를 사용하여 테이블을 지정할 수 있습니다. 대상 테이블이 결정되지 않은 경우 **Merge LOAD** 구조는 새 테이블에 저장

됩니다.

Concatenate 접두사가 사용된 경우 결과 테이블에는 기존 테이블과 병합할 입력의 합집합에 해당하는 필드 집합이 있습니다. 따라서 대상 테이블에는 병합할 입력으로 사용되는 변경 로그보다 더 많은 필드가 있을 수 있습니다.

부분 로드도 전체 로드와 동일합니다. 한 가지 차이점은 부분 로드의 경우 새 테이블을 거의 만들지 않는다는 점입니다. **Only** 절을 사용하지 않은 경우 이전 스크립트 실행과 동일한 필드 집합이 있는 대상 테이블이 항상 존재합니다.

시퀀스 번호

로드된 변경 로그가 누적된 로그인 경우, 즉 이미 로드된 변경 사항이 포함된 경우 매개 변수 SequenceNoVar을 **Where** 절에서 사용하여 입력 데이터의 양을 제한할 수 있습니다. 그런 다음 필드 SequenceNoField가 SequenceNoVar보다 큰 레코드만 로드하도록 **Merge LOAD**를 만들 수 있습니다. 완료되면 **Merge LOAD**가 SequenceNoField 필드에 표시된 최댓값으로 SequenceNoVar에 새 값을 할당합니다.

작업

Merge LOAD는 대상 테이블보다 필드를 적게 포함할 수 있습니다. 작업마다 누락된 필드를 다르게 처리힙니다.

Insert: **Merge LOAD**에 누락되었지만 대상 테이블에 있는 필드는 대상 테이블에 NULL을 가져옵니다.

삭제: 누락된 필드는 결과에 영향을 미치지 않습니다. 관련 레코드는 삭제됩니다.

Update: **Merge LOAD**에 나열된 필드는 대상 테이블에서 업데이트됩니다. 누락된 필드는 변경되지 않습니다. 즉, 다음 두 문은 동일하지 않습니다.

- 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 ...;

첫 번째 문은 나열된 레코드를 업데이트하고 F2를 NULL로 변경합니다. 두 번째는 F2를 변경하지 않는 대신 대상 테이블의 값을 유지합니다.

예

예 1: 지정된 테이블과 간단한 병합

이 예에서 Persons라고 명명된 인라인 테이블은 3개의 행으로 로드됩니다. 그런 다음 **Merge**를 통해 테이블이 다음과 같이 변경됩니다.

- *Mary, 4* 행을 추가합니다.
- *Steven, 3* 행을 삭제합니다.
- *Jake*에 숫자 *5*를 할당합니다.

LastChangeDate 변수는 **Merge**가 실행된 후 *ChangeDate* 열의 최댓값으로 설정됩니다.

로드 스크립트

예제 스크립트를 앱에 추가하고 실행합니다. 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.

```
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
];
```

결과

Merge Load 전에 결과 테이블은 다음과 같이 나타납니다.

Resulting table

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

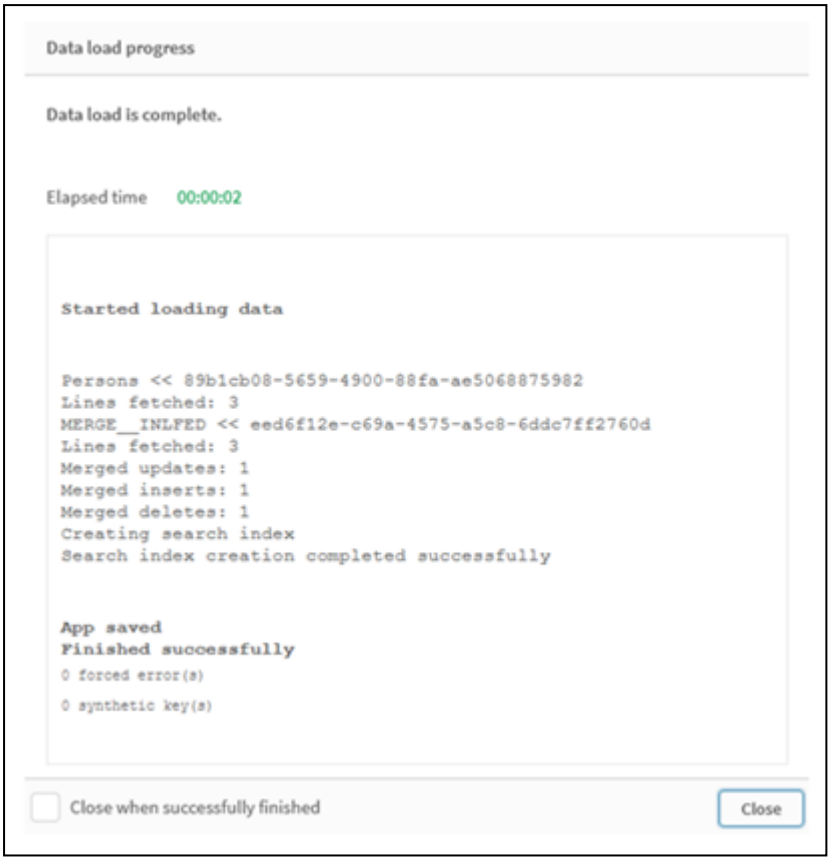
Merge Load의 결과 테이블은 다음과 같이 나타납니다.

Resulting table

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

데이터가 로드되면 **데이터 로드 진행률** 대화 상자에 수행되는 작업이 표시됩니다.

데이터 로드 진행률 대화 상자



예 2: 누락된 필드가 있는 데이터 로드 스크립트

이 예에서는 위와 동일한 데이터가 로드되지만 이제 각 개인의 ID가 포함되어 있습니다.

Merge를 통해 테이블이 다음과 같이 변경됩니다.

- *Mary, 4* 행을 추가합니다.
- *Steven, 3* 행을 삭제합니다.
- *Jake*에 숫자 *5*를 할당합니다.
- *Jill*에 숫자 *6*을 할당합니다.

로드 스크립트

여기서는 두 개의 **Merge Load** 문을 사용합니다. 하나는 'Insert'와 'Delete'에 대한 것이고 또 하나는 'Update' 입에 대한 것니다.

예제 스크립트를 앱에 추가하고 실행합니다. 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.

```
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
];
```

결과

Merge Load 문의 결과 테이블은 다음과 같이 나타납니다.

Resulting table

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

두 번째 **Merge** 문은 **Name** 필드를 포함하지 않으므로이름이 변경되지 않았습니다.

예 3: 데이터 로드 스크립트 - ChangeDate와 Where-절을 사용한 부분 로드

다음 예에서 **Only** 인수는 **Merge** 명령이 부분 로드 중에만 실행되도록 지정합니다. 업데이트는 이전에 캡처된 LastChangeDate를 기반으로 필터링됩니다. **Merge**가 완료되면 LastChangeDate 변수에 병합 중에 처리된 ChangeDate 열의 최댓값이 할당됩니다.

로드 스크립트

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

NoConcatenate

NoConcatenate 접두사를 사용하면 동일한 필드 집합을 포함한 두 개의 로드된 테이블이 두 개의 개별 내부 테이블로 처리됩니다. 그렇지 않은 경우 이 테이블은 자동으로 연결됩니다.

구문:

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

기본적으로 스크립트에서 이전에 로드된 테이블과 동일한 수의 필드 및 일치하는 필드 이름을 포함하는 테이블이 로드되면 Qlik Sense는 이 두 테이블을 자동으로 연결합니다. 이는 두 번째 테이블의 이름이 다르게 지정되더라도 발생합니다.

그러나 두 번째 테이블의 LOAD 문이나 select 문 앞에 스크립트 접두사 NoConcatenate가 포함되어 있으면 이 두 테이블이 별도로 로드됩니다.

NoConcatenate의 일반적인 사용 사례는 원본 데이터의 복사본을 유지하면서 해당 복사본에서 일부 임시 변환을 수행하기 위해 테이블의 임시 복사본을 만들어야 하는 경우입니다. NoConcatenate는 소스 테이블에 암시적으로 다시 추가하지 않고 해당 복사본을 만들 수 있는지 확인합니다.

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

함수 예

예	결과
Source: LOAD A,B from file1.csv; CopyOfSource: NoConcatenate LOAD A,B resident Source;	A 및 B를 측정값으로 사용하는 테이블이 로드됩니다. 동일한 필드를 가진 두 번째 테이블은 NoConcatenate 변수를 사용하여 별도로 로드됩니다.

예 1 - 암시적 연결

로드 스크립트 및 결과

개요

이 예에서는 두 개의 로드 스크립트를 순차적으로 추가합니다.

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Transactions라는 테이블로 전송되는 날짜 및 금액이 포함된 초기 데이터 집합.

첫 번째 로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- id
- date
- amount

첫 번째 결과 테이블

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

두 번째 로드 스크립트

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 동일한 필드가 있는 두 번째 데이터 집합이 Sales라는 테이블로 전송됩니다.

```
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
];
```

결과

데이터를 로드하고 테이블로 이동합니다.

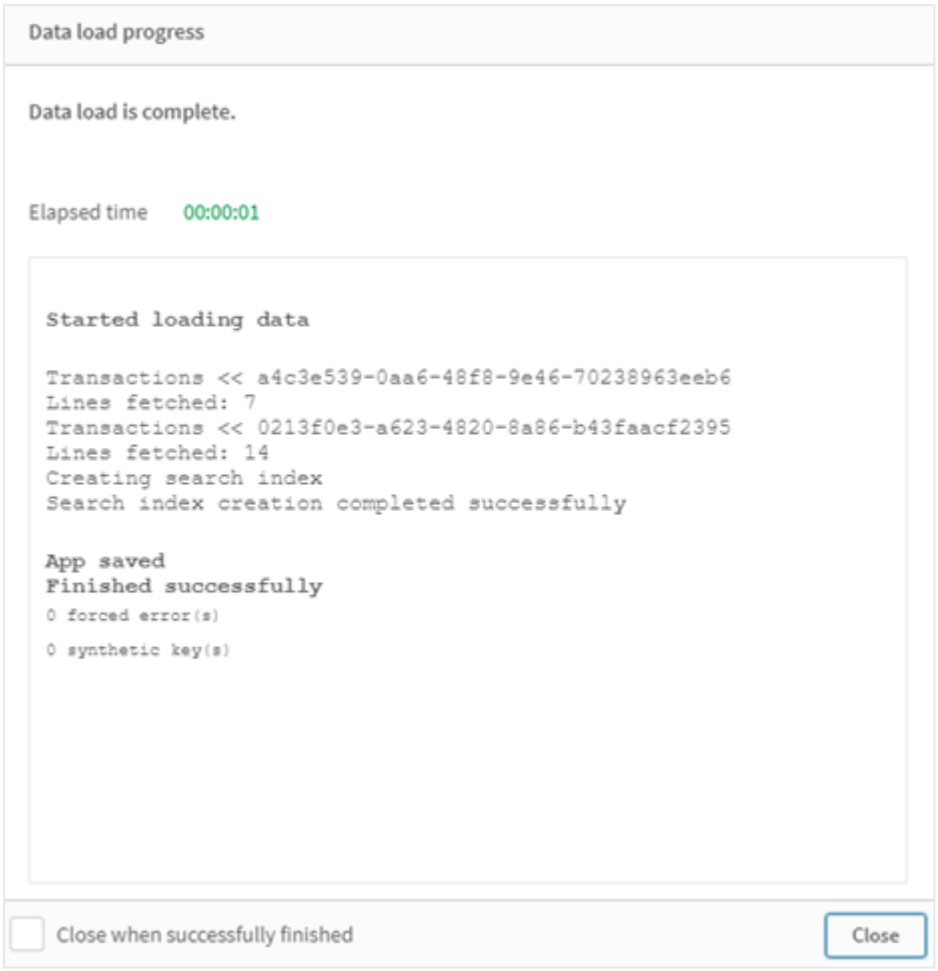
두 번째 결과 테이블

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

스크립트가 실행되면 동일한 필드 이름의 동일한 필드 수를 공유하는 두 데이터 집합으로 인해 `Sales` 테이블이 기존 `Transactions` 테이블에 암시적으로 연결됩니다. 이는 결과 집합 `'Sales'`의 이름을 지정하려고 시도하는 두 번째 테이블 이름 태그에도 불구하고 발생합니다.

데이터 로드 진행률 로그를 보면 Sales 데이터 집합이 암시적으로 연결되어 있음을 알 수 있습니다.

암시적으로 연결되는 트랜잭션 데이터를 보여 주는 데이터 로드 진행 로그.



예 2 – 사용 사례 시나리오

로드 스크립트 및 결과

개요

이 사용 사례 시나리오에는 다음이 있습니다.

- 다음이 포함된 트랜잭션 데이터 집합:
  - id
  - date
  - amount(GBP)
- 다음이 포함된 통화 테이블:
  - USD에서 GBP로의 변환율
- 다음이 포함된 두 번째 트랜잭션 데이터 집합:
  - id

- date
- amount(USD)

5개의 스크립트를 순차적으로 로드합니다.

- 첫 번째 로드 스크립트에는 Transactions라는 테이블로 전송되는 GBP 단위의 날짜와 금액이 포함된 초기 데이터 집합이 포함되어 있습니다.
- 두 번째 로드 스크립트에는 다음이 포함됩니다.
    - Transactions_in_USD라는 테이블로 전송되는, 날짜와 금액(USD)이 있는 두 번째 데이터 집합.
    - 암시적 연결을 방지하기 위해 Transactions_in_USD 데이터 집합의 LOAD 문 앞에 배치되는 noconcatenate 접두사.
- 세 번째 로드 스크립트에는 Transactions_in_USD 테이블에서 GBP와 USD 간의 통화 환율을 만드는 데 사용할 join 접두사가 포함되어 있습니다.
- 네 번째 로드 스크립트에는 초기 Transactions 테이블에 Transactions_in_USD를 추가할 concatenate 접두사가 포함되어 있습니다.
- 다섯 번째 로드 스크립트에는 데이터가 Transactions 테이블에 연결된 Transactions_in_USD 테이블을 제거하는 drop table 문이 포함되어 있습니다.

첫 번째 로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- id
- date
- amount

첫 번째 로드 스크립트 결과

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

id	date	amount
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

테이블에서 금액(GBP)이 있는 초기 데이터 집합을 보여 줍니다.

두 번째 로드 스크립트

```
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
];
```

결과

데이터를 로드하고 테이블로 이동합니다.

두 번째 로드 스크립트 결과

id	date	amount
1	12/30/2018	23.56
2	12/07/2018	556.31
3	12/16/2018	5.75
4	12/22/2018	125.00
5	12/22/2018	484.21
6	12/22/2018	59.18
7	12/23/2018	177.42
8	01/01/2019	164.27
9	01/03/2019	384.00
10	01/06/2019	25.82
11	01/09/2019	312.00

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

Transactions_in_USD 테이블의 두 번째 데이터 집합이 추가된 것을 볼 수 있습니다.

세 번째 로드 스크립트

이 로드 스크립트는 USD에서 GBP로의 환율을 Transactions_in_USD 테이블에 조인합니다.

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

결과

데이터를 로드하고 데이터 모델 뷰어로 이동합니다. Transactions_in_USD 테이블을 선택하면 모든 기존 레코드에 0.7의 'rate' 필드 값이 있음을 알 수 있습니다.

네 번째 로드 스크립트

Resident LOAD를 사용하여 이 로드 스크립트는 금액을 USD로 변환한 후 Transactions_in_USD 테이블을 Transactions 테이블에 연결합니다.

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

결과

데이터를 로드하고 테이블로 이동합니다. 8행에서 14행까지 금액(GBP)이 포함된 새 항목이 표시됩니다.

네 번째 로드 스크립트 결과

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

id	date	amount
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

다섯 번째 로드 스크립트

이 로드 스크립트는 네 번째 로드 스크립트 결과 테이블에서 중복 항목을 삭제하고 금액이 GBP인 항목만 남깁니다.

```
drop tables Transactions_in_USD;
```

결과

데이터를 로드하고 테이블로 이동합니다.

다섯 번째 로드 스크립트 결과

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

id	date	amount
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

다섯 번째 로드 스크립트를 로드한 후 결과 테이블에는 두 트랜잭션 데이터 집합에 있는 14개의 트랜잭션이 모두 표시됩니다. 그러나 트랜잭션 8-14의 금액은 GBP로 변환되었습니다.

두 번째 로드 스크립트에서 Transactions_in_USD 앞에 사용된 NoConcatenate 접두사를 제거하면 "테이블 'Transactions_in_USD'를 찾을 수 없음" 오류와 함께 스크립트가 실패합니다. 이는 Transactions_in_USD 테이블이 원본 Transactions 테이블에 자동으로 연결되었기 때문입니다.

Only

Only 스크립트 키워드는 집계 함수로 사용되거나 부분 다시 로드 접두사 **Add**, **Replace** 및 **Merge**에서 구문의 일부로 사용됩니다.

Outer

명시적인 **Join** 접두사 앞에 **Outer** 접두사를 사용하여 외부 조인을 지정할 수 있습니다. 외부 조인 시 두 테이블 간의 모든 조합이 생성됩니다. 따라서 결과 테이블은 원시 데이터 테이블의 필드 값 조합을 포함하며 연결 필드 값은 한 테이블이나 두 테이블 모두에 표시됩니다. **Outer** 키워드는 옵션이며 join 접두사를 지정하지 않을 경우 사용되는 기본 조인 유형입니다.

구문:

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

인수:

인수

인수	설명
tablename	로드된 테이블과 비교할 명명된 테이블입니다.
loadstatement또는 selectstatement	로드된 테이블에 사용되는 **LOAD** 또는 **SELECT** 문입니다.

예

로드 스크립트

예제 스크립트를 앱에 추가하고 실행합니다. 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.

```
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 ];
```

결과 테이블

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

설명

이 예에서 두 테이블(Table1 및 Table2)은 Table1이라는 레이블이 붙은 단일 테이블로 병합됩니다. 이와 같은 경우 **outer** 접두사는 여러 테이블을 단일 테이블로 조인하여 단일 테이블의 값에 대한 집계를 수행하는 데 자주 사용됩니다.

부분 로드

전체 다시 로드는 항상 기존 데이터 모델의 모든 테이블을 삭제하여 시작한 다음 로드 스크립트를 실행합니다.

부분 로드는 이 작업을 수행하지 않습니다. 대신 데이터 모델의 모든 테이블을 유지한 다음 앞에 **Add**, **Merge** 또는 **Replace** 접두사가 붙은 **Load** 및 **Select** 문만 실행합니다. 다른 데이터 테이블은 명령의 영향을 받지 않습니다. **only** 인수는 문이 부분 로드 중에만 실행되어야 하며 전체 로드 중에는 무시되어야 함을 나타냅니다. 다음 테이블에는 부분 및 전체 로드에 대한 문 실행이 요약되어 있습니다.

문	전체 로드	부분 로드
Load ...	문이 실행됨	문이 실행되지 않음
Add/Replace/Merge Load ...	문이 실행됨	문이 실행됨
Add/Replace/Merge Only Load ...	문이 실행되지 않음	문이 실행됨

부분 다시 로드는 전체 다시 로드와 비교하여 몇 가지 이점이 있습니다.

- 최근에 변경된 데이터만 로드하면 되므로 더 빠릅니다. 큰 데이터 집합의 경우 차이가 상당합니다.
- 로드되는 데이터가 적기 때문에 메모리가 덜 소모됩니다.
- 소스 데이터에 대한 쿼리가 더 빠르게 실행되어 네트워크 문제의 위험이 줄어들기 때문에 더 안정적입니다.

부분 다시 로드가 제대로 작동하려면 부분 다시 로드가 트리거되기 전에 앱을 데이터와 함께 열어야 합니다.

로드 버튼을 사용하여 부분 로드를 수행합니다. Qlik Engine JSON API을 사용할 수도 있습니다.

제한 사항

전체 다시 로드 중에 존재했지만 부분 다시 로드 중에는 존재하지 않은 테이블에 대한 참조가 있는 명령이 있는 경우 부분 다시 로드가 실패합니다.

예

명령 예

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

여기서 <Table_removed_after_full_reload>는 전체 다시 로드에는 존재했지만 부분 다시 로드에는 존재하지 않는 테이블입니다.

해결책

해결책으로 명령을 다음 if 문으로 묶을 수 있습니다.

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

부분 로드는 데이터에서 값을 제거할 수 있습니다. 그러나 내부적으로 유지 관리되는 테이블인 고유 값 목록에는 반영되지 않습니다. 따라서 부분 로드 후 목록에는 마지막 전체 로드 이후 필드에 존재했던 모든 고유 값이 포함되며, 이는 부분 로드 이후에 현재 존재하는 것보다 많을 수 있습니다. 이는 FieldValueCount() 및 FieldValue() 함수의 출력에 영향을 줍니다. FieldValueCount()는 잠재적으로 현재 필드 값 수보다 큰 수를 반환할 수 있습니다.

예

예 1

로드 스크립트

이 스크립트 예를 앱에 추가하고 부분 다시 로드를 수행합니다. 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.

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

결과

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

설명

문은 부분 로드 중에만 실행됩니다. "고유" 접두사가 생략된 경우 각 후속 부분 로드에서 **Num** 필드의 개수가 증가합니다.

예 2

로드 스크립트

이 예제 스크립트를 앱에 추가합니다. 전체 로드를 수행하고 결과를 확인합니다. 다음으로, 부분 로드를 수행하고 결과를 확인합니다. 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.

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

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

결과

Output table after full reload

ID	Value
1	1
2	2
3	3
4	4
5	5

ID	Value
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

설명

첫 번째 테이블은 전체 로드 중에 로드되고 두 번째 테이블은 부분 로드 중에 첫 번째 테이블을 대체합니다.

Replace

Replace 스크립트 키워드는 문자열 함수로 사용되거나 부분 재로드에서 접두사로 사용됩니다.

Replace

Replace 접두사를 스크립트의 **LOAD** 또는 **SELECT** 문에 추가하여 로드된 테이블이 다른 테이블을 대체하도록 지정할 수 있습니다. 또한 이 문을 부분 로드에서 실행하도록 지정합니다. **Replace** 접두사는 **Map** 문에도 사용할 수 있습니다.

부분 다시 로드가 제대로 작동하려면 부분 다시 로드가 트리거되기 전에 앱을 데이터와 함께 열어야 합니다.

로드 버튼을 사용하여 부분 로드를 수행합니다. Qlik Engine JSON API을 사용할 수도 있습니다.

구문:

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

```
Replace [only] mapstatement
```

일반(부분 아님) 로드 중에는 **ReplaceLOAD** 구조가 일반 **LOAD** 문으로 작동하지만 **Drop Table**이 앞에 와야 합니다. 먼저 이전 테이블이 삭제된 다음 레코드가 생성되어 새 테이블로 저장됩니다.

Concatenate 접두사가 사용되거나 동일한 필드 집합의 테이블이 있는 경우 이는 삭제할 관련 테이블입니다. 그렇지 않으면 삭제할 테이블이 없으며 **Replace LOAD** 구조는 일반 **LOAD**와 동일합니다.

부분 로드도 같습니다. 유일한 차이는 항상 삭제할 이전 스크립트 실행의 테이블이 있다는 것입니다. **Replace LOAD** 구조는 항상 이전 테이블을 먼저 삭제한 다음 새 테이블을 만듭니다.

Replace Map...Using 문을 사용하면 부분 스크립트 실행 도중에도 매핑이 일어나게 됩니다.

인수:

인수

인수	설명
only	문이 부분 로드 중에만 실행되어야 함을 표시하는 선택적 한정자입니다. 일반(부분 아님) 로드 중에는 무시되어야 합니다.

예 및 결과:

예	결과
Tab1: Replace LOAD * from File1.csv;	일반 및 부분 재로드 도중에 Qlik Sense 테이블 Tab1이 먼저 삭제됩니다. 그 후 File1.csv에서 새 데이터가 로드되고 Tab1에 저장됩니다.
Tab1: Replace only LOAD * from File1.csv;	일반 재로드 도중에 이 문은 무시됩니다. 부분 재로드 도중에는 이전에 Tab1로 명명된 모든 Qlik Sense 테이블이 먼저 삭제됩니다. 그 후 File1.csv에서 새 데이터가 로드되고 Tab1에 저장됩니다.
Tab1: LOAD a,b,c from File1.csv; Replace LOAD a,b,c from File2.csv;	일반 재로드 도중에는 먼저 File1.csv 파일이 Qlik Sense 테이블 Tab1에 로드된 후 곧바로 삭제되고 File2.csv에서 로드된 새 데이터로 대체됩니다. File1.csv의 모든 데이터를 잃게 됩니다. 부분 재로드 도중에는 Qlik Sense 테이블 Tab1 전체가 먼저 삭제됩니다. 그 후 File2.csv에서 로드된 새 데이터로 대체됩니다.
Tab1: LOAD a,b,c from File1.csv; Replace only LOAD a,b,c from File2.csv;	일반 재로드 중에 데이터가 File1.csv에서 로드되어 Qlik Sense 테이블 Tab1에 저장됩니다. File2.csv는 무시됩니다. 부분 재로드 도중에는 Qlik Sense 테이블 Tab1 전체가 먼저 삭제됩니다. 그 후 File2.csv에서 로드된 새 데이터로 대체됩니다. File1.csv의 모든 데이터를 잃게 됩니다.

Right

Join 및 **Keep** 접두사 앞에 **right** 접두사를 사용할 수 있습니다.

join 앞에 사용할 경우 오른쪽 조인을 사용하도록 지정됩니다. 결과 테이블은 원시 데이터 테이블의 필드 값 조합만을 포함하며 연결 필드 값은 두 번째 테이블에 표시됩니다. **keep** 앞에 사용할 경우 Qlik Sense에 저장되기 전에 첫 번째 원시 데이터 테이블을 두 번째 테이블과의 공통 교집합으로 축소하도록 지정됩니다.

같은 이름의 문자열 함수를 찾고 계십니까? 참조: Right (page 1413)

구문:

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

인수:

인수

인수	설명
tablename	로드된 테이블과 비교할 명명된 테이블입니다.
loadstatement또는 selectstatement	로드된 테이블에 사용되는 **LOAD** 또는 **SELECT** 문입니다.

예

로드 스크립트

예제 스크립트를 앱에 추가하고 실행합니다. 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.

```
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 ];
```

결과

결과 테이블

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

설명

이 예에서는 두 번째(오른쪽) 테이블에 있는 값만 조인되는 오른쪽 조인 출력을 보여 줍니다.

Sample

LOAD 또는 **SELECT** 문의 **sample** 접두사는 데이터 소스에서 무작위 레코드 샘플을 로드하는 데 사용됩니다.

구문:

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

평가되는 표현식은 Qlik Sense 응용 프로그램에 로드될 데이터 집합의 레코드 비율을 정의하는 것이 아니라 읽은 각 레코드가 응용 프로그램에 로드될 확률을 정의합니다. 즉, `p = 0.5` 값을 지정한다고 해서 전체 레코드 수의 50%가 로드되는 것이 아니라 각 레코드에 대해 Qlik Sense 응용 프로그램에 로드될 확률이 50%가 된다는 것을 의미합니다.

인수

인수	설명
p	0보다 크고 1보다 작거나 같은 숫자로 평가되는 임의의 표현식입니다. 이 숫자는 지정된 레코드가 읽힐 확률을 나타냅니다. 모든 레코드를 읽을 수 있지만 그 중 일부만 Qlik Sense에 로드됩니다.

사용 시기

샘플은 데이터, 분포 또는 필드 콘텐츠의 특성을 이해하기 위해 큰 테이블에서 가져온 데이터를 샘플링하려는 경우에 유용합니다. 데이터의 하위 집합을 가져오기 때문에 데이터 로드가 더 빨라져 스크립트를 더 빠르게 테스트할 수 있습니다. `First`와 달리 `Sample` 함수는 처음 몇 행에 국한되지 않고 전체 테이블에서 데이터를 가져옵니다. 이는 경우에 따라 데이터를 보다 정확하게 표시할 수 있습니다.

다음 예에서는 `Sample` 스크립트 접두사의 두 가지 가능한 사용을 보여 줍니다.

```
Sample 0.15 SQL SELECT * from Longtable;

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

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 `SET DateFormat` 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

예 1 – 인라인 테이블의 샘플

로드 스크립트 및 결과

개요

이 예에서 스크립트는 7개의 레코드가 포함된 데이터 집합의 샘플 데이터 집합을 인라인 테이블의 Transactions라는 테이블로 로드합니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- id
- amount

다음 측정값을 추가합니다.

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

결과 테이블

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

이 예에서 사용된 로드의 반복에서 7개의 레코드를 모두 읽었지만 데이터 테이블에는 4개의 레코드만 로드되었습니다. 재실행 로드로 인해 다른 수와 다른 레코드 집합이 응용 프로그램에 로드될 수 있습니다.

예 2 – 자동 생성된 테이블의 샘플

로드 스크립트 및 결과

개요

이 예에서 Autogenerate를 사용하여 date, id 및 amount 필드가 있는 100개 레코드의 데이터 집합이 만들어집니다. 그러나 값이 0.1인 Sample 접두사가 사용됩니다.

로드 스크립트

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

Autogenerate(100);
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- id
- amount

다음 측정값을 추가합니다.

결과 테이블

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

이 예에서 로드 사용을 반복하여 만들어진 데이터 집합에서 7개의 레코드가 로드되었습니다. 다시 한 번, 모든 재실행 로드로 인해 다른 수와 다른 레코드 집합이 응용 프로그램에 로드될 수 있습니다.

Semantic

semantic 로드 접두사는 Qlik Sense에서 트리 구조, 자체 참조 부모-자식으로 구조화된 데이터 및/또는 그래프로 설명할 수 있는 데이터와 같은 관계형 데이터를 연결하고 관리하는 데 사용할 수 있는 특수한 유형의 필드를 만듭니다.

semantic 로드는 *Hierarchy (page 62)* 및 *HierarchyBelongsTo (page 64)* 접두사와 유사하게 작동할 수 있습니다. 세 접두사 모두 관계형 데이터를 탐색하기 위한 효과적인 프런트 엔드 솔루션의 빌딩 블록으로 사용할 수 있습니다.

구문:

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

의미 체계 로드는 아래 테이블에 표시된 것처럼 순서가 지정된 각 필드가 나타내는 항목에 대한 엄격한 정의와 함께 정확히 3개 또는 4개의 필드 너비의 입력을 예상합니다.

의미 체계 로드 필드

필드 이름	필드 설명
첫 번째 필드:	이 태그는 관계가 있는 두 개체 중 첫 번째 개체를 나타냅니다.
두 번째 필드:	이 태그는 첫 번째 개체와 두 번째 개체 간의 "정방향" 관계를 설명하는 데 사용됩니다. 첫 번째 개체가 자식이고 두 번째 개체가 부모인 경우 자식에서 부모로의 관계를 따르는 것처럼 "parent" 또는 "parent of"를 나타내는 관계 탭을 만들 수 있습니다.
세 번째 필드:	이 태그는 관계가 있는 두 개체 중 두 번째 개체를 나타냅니다.
네 번째 필드:	이 필드는 선택 사항입니다. 이 태그는 첫 번째 개체와 두 번째 개체 간의 "역방향" 또는 "역" 관계를 설명합니다. 첫 번째 개체가 자식이고 두 번째 개체가 부모인 경우 관계 탭에는 부모에서 자식으로의 관계를 따르는 것처럼 "child" 또는 "child of"가 표시될 수 있습니다. 네 번째 필드를 추가하지 않으면 두 번째 필드 태그가 어느 방향으로든 관계를 설명하는 데 사용됩니다. 이 경우 화살표 기호가 태그의 일부로 자동으로 추가됩니다.

다음 코드는 semantic 접두사의 예입니다.

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

세 번째 필드에 첫 번째 필드와 동일한 레이블을 지정하는 것이 허용되고 일반적인 사례입니다. 이렇게 하면 자체 참조 조회가 만들어져 관련 개체와 한 번에 한 관계 단계씩 멀어지는 개체를 따를 수 있습니다. 세 번째 필드에 동일한 이름이 없으면 최종 결과는 개체에서 한 단계만 멀어진 직접 관계 이웃까지의 간단한 조회가 될 것이며, 이는 실제로 거의 사용하지 않는 출력입니다.

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

관련 함수

함수	상호 작용
Hierarchy (page 62)	계층 로드 접두사는 부모-자식 및 기타 그래프와 같은 데이터 구조의 노드를 분할 및 구성하고 테이블로 변환하는 데 사용됩니다.
HierarchyBelongsTo (page 64)	HierarchyBelongsTo 로드 접두사는 부모-자식 및 기타 그래프와 같은 데이터 구조의 상위를 찾아 구성하고 테이블로 변환하는 데 사용됩니다.

예 - 의미 체계 접두사를 사용하여 관계를 연결하기 위한 특수 필드 만들기

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- GeographyTree라는 테이블에 로드되는 지리 관계 레코드를 나타내는 데이터 집합.
    - 각 항목에는 줄 시작 부분에 ID가 있고 줄 끝 부분에 ParentID가 있습니다.
- Relation으로 레이블이 지정된 하나의 특수 동작 필드를 추가하는 semantic 접두사.

로드 스크립트

```
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;
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- Id
- Geography

그런 다음 차원으로 Relation을 사용하여 필터 창을 만듭니다. **편집 완료**를 클릭합니다.

결과 테이블

Id	Geography
1	World
2	Europe
3	Asia
4	North America
5	South America
6	UK
7	Germany
8	Sweden
9	South Korea

Id	Geography
10	North Korea
11	China
12	London
13	Birmingham

필터 창

관계

자식

부모

테이블의 Geography 차원에서 **Europe**을 클릭하고 필터 창의 Relation 차원에서 **자식**을 클릭합니다. 테이블의 예상 결과를 확인합니다.

Europe의 "자식"을 보여 주는 결과 테이블

Id	Geography
6	UK
7	Germany
8	Sweden

자식을 다시 클릭하면 한 단계 더 아래로 UK의 "자식"인 장소가 표시됩니다.

UK의 "자식"을 표시하는 결과 테이블

Id	Geography
12	London
13	Birmingham

Unless

unless 접두사 및 접미사는 문 또는 exit 절을 평가할지 여부를 결정하는 조건절을 만드는 데 사용됩니다. 전체 **if..end if** 문의 간단한 대체 옵션으로 사용되는 경우도 있습니다.

구문:

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

statement 또는 **exitstatement**는 **condition**이 False로 평가된 경우에만 실행됩니다.

unless 접두사는 추가적인 **when** 또는 **unless** 접두사를 포함한 하나 이상의 다른 문을 이미 가지고 있는 문에서도 사용할 수 있습니다.

인수

인수	설명
condition	True 또는 False로 평가되는 논리 표현식입니다.
statement	제어 문을 제외한 Qlik Sense 스크립트 문입니다.
exitstatement	**exit for**, **exit do** 또는 **exit sub** 절 혹은 **exit script** 문.

사용 시기

Unless 문은 부울 결과를 반환합니다. 일반적으로 이 유형의 함수는 사용자가 스크립트의 일부를 조건부로 로드하거나 제외하려는 경우 조건으로 사용됩니다.

다음 줄에서는 Unless 함수 사용 방법에 대한 세 가지 예를 보여 줍니다.

```
exit script unless A=1;

unless A=1 LOAD * from myfile.csv;

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

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

예 1 – Unless 접두사

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 값이 1로 지정된 변수 A 만들기
- 변수 A = 2가 아닌 경우 Transactions라는 테이블에 로드되는 데이터 집합.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- id
- date
- amount

결과 테이블

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

변수 A는 스크립트 시작 시 값 1을 할당받았으므로 Unless 접두사 뒤에 오는 조건을 평가하여 결과 FALSE를 반환합니다. 결과적으로 스크립트는 Load 문을 계속 실행합니다. 결과 테이블에서 Transactions 테이블의 모든 레코드를 볼 수 있습니다.

이 변수 값을 2로 설정하면 데이터 모델에 데이터가 로드되지 않습니다.

예 2 – Unless 접미사

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트는 초기 데이터 집합을 Transactions라는 테이블에 로드하는 것으로 시작합니다. 그런 다음 Transactions 테이블에 10개 미만의 레코드가 없으면 스크립트가 종료됩니다.

이 조건으로 인해 스크립트가 종료되지 않으면 추가 트랜잭션 집합이 Transactions 테이블에 연결되고 이 프로세스가 반복됩니다.

로드 스크립트

```
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 ;
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- id
- date
- amount

결과 테이블

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

로드 스크립트의 3개 데이터 집합 각각에는 7개의 레코드가 있습니다.

첫 번째 데이터 집합(트랜잭션 id 1~7 포함)이 응용 프로그램에 로드됩니다. Unless 조건은 Transactions 테이블에 10개 미만의 행이 있는지 여부를 평가합니다. 이는 TRUE로 평가되므로 두 번째 데이터 집합(트랜잭션 id 8~14 포함)이 응용 프로그램에 로드됩니다. 두 번째 Unless 조건은 Transactions 테이블에 10개 미만의 레코드가 있는지 평가합니다. 이는 FALSE로 평가되므로 해당 스크립트가 종료됩니다.

예 3 – 다중 Unless 접두사

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

이 예에서는 하나의 트랜잭션을 포함하는 데이터 집합이 Transactions라는 테이블로 만들어집니다. 그런 다음 두 개의 중첩된 unless 문이 다음을 평가하는 'for' 루프가 트리거는 경우는 다음과 같습니다.

1. Transactions 테이블에 100개보다 많은 레코드가 없는 경우
2. Transactions 테이블의 레코드 수가 6의 배수가 아닌 경우

이러한 조건이 FALSE이면 추가로 7개의 레코드가 생성되어 기존 Transactions 테이블에 연결됩니다. 이 프로세스는 두 트랜잭션 중 하나가 값 TRUE를 반환할 때까지 반복됩니다.

로드 스크립트

```
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
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다.id.

결과 테이블

id
0
1
2
3
4
5
+30 추가 행

'for' 루프에서 발생하는 중첩된 unless 문은 다음을 평가합니다.

1. Transactions 테이블에 100개보다 많은 행이 있습니까?
2. Transactions 테이블의 총 레코드 수가 6의 배수입니까?

두 unless 문 모두 값 FALSE를 반환할 때마다 추가로 7개의 레코드가 생성되어 기존 Transactions 테이블에 연결됩니다.

이러한 명령문은 값 FALSE를 5번 반환하며, 이때 Transactions 테이블에는 총 36개의 데이터 행이 있습니다.

그 후, 두 번째 Unless 문은 값 TRUE를 반환하므로 다음에 오는 LOAD 문은 더 이상 실행되지 않습니다.

When

when 접두사 및 접미사는 문 또는 exit 절을 실행할지 여부를 결정하는 조건절을 만드는 데 사용됩니다. 전체 **if..end if** 문의 간단한 대체 옵션으로 사용되는 경우도 있습니다.

구문:

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

반환 데이터 유형: 부울

Qlik Sense에서 부울 true 값은 -1로 표시되고 false 값은 0으로 표시됩니다.

statement 또는 **exitstatement**는 조건이 TRUE로 평가된 경우에만 실행됩니다.

When 접두사는 추가적인 When 또는 Unless 접두사를 포함한 하나 이상의 다른 문을 이미 가지고 있는 문에서도 사용할 수 있습니다.

사용 시기

When 문은 부울 결과를 반환합니다. 일반적으로 이러한 유형의 함수는 사용자가 스크립트의 일부를 로드하거나 제외하려는 경우 조건으로 사용됩니다.

인수

인수	설명
condition	TRUE 또는 FALSE로 평가되는 논리 표현식
statement	제어 문을 제외한 Qlik Sense 스크립트 문입니다.
exitstatement	**exit for**, **exit do** 또는 **exit sub** 절 혹은 **exit script** 문.

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

함수 예

예	결과
`exit script when A=1;`	`A=1` 문이 TRUE로 평가되면 스크립트가 중지됩니다.
`when A=1 LOAD * from myfile.csv;`	`A=1` 문이 TRUE로 평가되면 `myfile.csv`가 로드됩니다.
`when A=1 unless B=2 drop table Tab1;`	`A=1` 문이 TRUE로 평가되고 `B=2`가 FALSE로 평가되면 `Tab1` 테이블이 삭제됩니다.

예 1 – When 접두사

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 'Transactions'라는 테이블로 전송되는 날짜 및 금액이 포함된 데이터 집합.
- 변수 A가 만들어져 값 1을 갖는다는 Let 문.
- A가 1이면 스크립트가 계속 로드된다는 조건을 제공하는 When 조건.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- id
- date
- amount

결과 테이블

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

변수 A는 스크립트 시작 시 값 1을 할당받았으므로 when 접두사 뒤에 오는 조건을 평가하여 결과 TRUE를 반환합니다. TRUE 결과를 반환하므로 스크립트는 계속해서 LOAD 문을 실행합니다. 결과 테이블의 모든 레코드를 볼 수 있습니다.

이 변수 값이 1과 같지 않은 값으로 설정되면 데이터 모델에 데이터가 로드되지 않습니다.

예 2 – When 접미사

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 날짜와 금액이 포함된 3개의 데이터 집합이 'Transactions'라는 테이블로 전송됩니다.
  - 첫 번째 데이터 집합에는 트랜잭션 1-7이 포함됩니다.
  - 두 번째 데이터 집합에는 트랜잭션 8-14가 포함됩니다.
  - 세 번째 데이터 집합에는 트랜잭션 15-21이 포함됩니다.
- 'Transactions' 테이블에 10개보다 많은 행이 포함되어 있는지 여부를 확인하는 when 조건. when 문 중 하나라도 TRUE로 평가되면 로드 스크립트가 중지됩니다. 이 조건은 세 개의 데이터 집합 각각의 끝에 배치됩니다.

로드 스크립트

```
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 ;
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- id
- date
- amount

결과 테이블

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

3개의 데이터 집합 각각에는 7개의 트랜잭션이 있습니다. 첫 번째 데이터 집합에는 트랜잭션 1-7이 포함되어 있으며 응용 프로그램에 로드됩니다. 이 LOAD 문 다음의 when 조건은 'Transactions' 테이블에 10개 미만의 행이 있으므로 FALSE로 평가됩니다. 로드 스크립트는 다음 데이터 집합으로 계속됩니다.

두 번째 데이터 집합에는 트랜잭션 8-14가 포함되어 있으며 응용 프로그램에 로드됩니다. 두 번째 When 조건은 'Transactions' 테이블에 10개보다 많은 행이 있으므로 TRUE로 평가됩니다. 따라서 스크립트가 종료됩니다.

예 3 - 다중 When 접두사

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 단일 트랜잭션을 포함하는 데이터 집합은 'Transactions'라는 테이블로 만들어집니다.
- 트리거되는 For 루프에는 다음을 평가하는 두 개의 중첩된 when 조건이 포함됩니다.

1. 'Transactions' 테이블에 100보다 적은 레코드가 있는지 평가
2. 'Transactions' 테이블의 레코드 수가 6의 배수가 아닌지 평가

로드 스크립트

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

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다.

- id

결과 테이블에는 처음 5개의 트랜잭션 ID만 표시되지만 로드 스크립트는 36개의 행을 만든 다음 When 조건이 충족되면 종료됩니다.

결과 테이블

id
0
1
2
3
4
5
+30 추가 행

For 루프의 중첩된 When 조건은 다음 질문을 평가합니다.

- 'Transactions' 테이블에 100개 미만의 행이 있습니까?
- 'Transactions' 테이블의 총 레코드 수가 6의 배수가 아닙니까?

두 When 조건 모두 TRUE 값을 반환할 때마다 추가로 7개의 레코드가 생성되어 기존 'Transactions' 테이블에 연결됩니다.

When 조건은 TRUE 값을 5번 반환합니다. 이 때 'Transactions' 테이블에는 총 36개의 데이터 행이 있습니다.

'Transactions' 테이블에 36행의 데이터가 만들어지면 두 번째 when 문은 FALSE 값을 반환하므로 다음에 오는 LOAD 문은 더 이상 실행되지 않습니다.

3.3 정규 스크립트 문

정규 문은 일반적으로 어떤 방식으로든 데이터를 편집하는 데 사용됩니다. 이러한 문은 스크립트에서 줄 수에 관계없이 작성할 수 있으며, 항상 세미콜론(";")으로 종결되어야 합니다.

모든 스크립트 키워드는 소문자와 대문자를 원하는 대로 조합하여 입력할 수 있습니다. 하지만 문에 사용되는 필드 및 변수 이름은 대/소문자가 구분됩니다.

정규 스크립트 문 개요

각 함수는 개요가 끝난 후에 더 자세히 설명합니다. 구문에서 함수 이름을 클릭하여 해당 함수에 대한 상세 설명에 즉시 액세스할 수도 있습니다.

Alias

alias 문은 이어지는 스크립트 내에서 발생할 때마다 필드 이름을 변경하는 데 사용할 별칭을 설정하는 데 사용됩니다.

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

Autonumber

이 문은 스크립트 실행 중에 발견된 필드의 평가된 각 고유 값에 해당하는 고유 정수 값을 생성합니다.

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

Binary

binary 문은 다른 QlikView 문서에서 Section Access 데이터를 포함한 데이터를 로드하는 데 사용됩니다.

```
Binary  [path] filename
```

comment

데이터베이스와 스프레드시트의 필드 주석(메타데이터)을 표시하는 방법을 제공합니다. 앱에 없는 필드 이름은 무시됩니다. 필드 이름이 여러 번 발견되는 경우 마지막 값이 사용됩니다.

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

comment table

데이터베이스나 스프레드시트의 테이블 주석(메타데이터)을 표시하는 방법을 제공합니다.

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

Connect

이 기능은 Qlik Sense SaaS에서 사용할 수 없습니다.

CONNECT 문은 OLE DB/ODBC 인터페이스를 통해 일반 데이터베이스에 대한 Qlik Sense 액세스를 정의하는 데 사용됩니다. ODBC의 경우, 먼저 ODBC 관리자를 사용하여 데이터 소스를 지정해야 합니다.

```
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

Declare 문은 필드 또는 함수 간의 관계를 정의할 수 있는 필드 정의를 만드는 데 사용됩니다. 일련의 필드 정의는 차원으로 사용할 수 있는 파생된 필드를 자동으로 생성하는 데 사용할 수 있습니다. 예를 들어, 캘린더 정의를 만들 수 있으며 이를 사용하여 날짜 필드에서 연도, 월, 주, 일 등의 관련 차원을 생성할 수 있습니다.

```
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

Derive 문은 **Declare** 문으로 만든 필드 정의를 바탕으로 파생된 필드를 생성하는 데 사용됩니다. 필드를 파생할 데이터 필드를 지정하거나 필드 태그를 바탕으로 명시적 또는 암시적으로 데이터 필드를 파생할 수 있습니다.

```
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

DIRECT QUERY 문을 사용하면 Direct Discovery 함수를 사용하여 ODBC 또는 OLE DB 연결을 통해 테이블에 액세스할 수 있습니다.

```
Direct Query [path]
```

Directory

Directory 문은 새 **Directory** 문이 만들어질 때까지 후속 **LOAD** 문에서 데이터 파일을 탐색할 디렉터리를 정의합니다.

```
Directory [path]
```

Disconnect

Disconnect 문은 현재 ODBC/OLE DB/사용자 지정 연결을 종료합니다. 이 문은 옵션입니다.

```
Disconnect
```

drop field
스크립트 실행 도중 언제든 **drop field** 문을 사용하여 데이터 모델, 즉 메모리에서 하나 또는 여러 개의 Qlik Sense 필드를 삭제할 수 있습니다. 테이블의 "distinct" 속성은 **drop field** 문 다음에 제거됩니다.

***drop field** 및 **drop fields**는 모두 허용되는 형식이며 결과에 차이가 없습니다. 테이블을 지정하지 않으면 필드가 발생되는 모든 테이블에서 삭제됩니다.*

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

drop table
스크립트 실행 도중 언제든 **drop table** 문을 사용하여 데이터 모델, 즉 메모리에서 하나 또는 여러 개의 Qlik Sense 내부 테이블을 삭제할 수 있습니다.

***drop table** 및 **drop tables** 형식을 둘 다 사용할 수 있습니다.*

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

Execute
Execute 문은 Qlik Sense가 데이터를 로드하는 동안 다른 프로그램을 실행하는 데 사용됩니다. 예를 들어, 필요한 규칙을 만드는 데 사용됩니다.

```
Execute commandline
```

FlushLog
FlushLog 문을 사용하면 Qlik Sense에서 스크립트 로그 파일에 스크립트 버퍼의 내용을 기록할 수 있습니다.

```
FlushLog
```

Force
force 문은 Qlik Sense가 후속 **LOAD** 및 **SELECT** 문의 필드 이름 및 필드 값을 대문자 또는 소문자로만 해석하거나 첫 글자를 항상 대문자로 해석하거나 원래대로(혼합) 해석하도록 지정합니다. 이 문을 사용하여 다른 규칙이 적용된 테이블의 필드 값을 연결할 수 있습니다.

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

LOAD
LOAD 문은 파일, 스크립트에 정의된 데이터, 이전에 로드한 테이블, 웹 페이지, 이후 **SELECT** 문의 결과에서 필드를 로드하거나 자동으로 데이터를 생성하여 필드를 로드합니다. 분석 연결에서 데이터를 로드할 수도 있습니다.

```
 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

let 문은 스크립트 변수 정의에 사용된다는 점에서 **set** 문과 차이가 있습니다. **let** 문은 **set** 문과는 반대로 변수에 할당하기 전 스크립트 런타임에 '='의 오른쪽에 있는 표현식을 평가합니다.

```
Let variablename=expression
```

Loosen Table

Loosen Table 문을 사용하면 스크립트 실행 중 하나 이상의 Qlik Sense 내부 데이터 테이블이 명시적으로 느슨하게 결합되도록 선언될 수 있습니다. 테이블이 느슨하게 결합된 경우 테이블 내 필드 값 간의 모든 연결이 제거됩니다. 느슨하게 결합된 테이블의 각 필드를 연결되지 않은 별도의 테이블로 로드하면 비슷한 효과를 낼 수 있습니다. 느슨한 결합은 테스트 중에 데이터 구조의 다른 부분을 임시로 격리할 때 유용할 수 있습니다. 느슨하게 결합된 테이블은 테이블 뷰어에서 점선으로 표시됩니다. 스크립트에서 **Loosen Table** 문을 하나 이상 사용하면 Qlik Sense가 스크립트 실행 전에 생성된 느슨하게 결합된 테이블 설정을 모두 무시하게 됩니다.

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

Map ... using

map ... using 문은 특정 필드 값 또는 표현식을 특정 매핑 테이블의 값에 매핑하는 데 사용됩니다. 매핑 테이블은 **Mapping** 문을 통해 만듭니다.

```
Map  *fieldlist Using  mapname
```

NullAsNull

NullAsNull 문은 이전에 **NullAsValue** 문으로 설정된 문자열 값으로의 NULL 값 변환을 해제합니다.

```
NullAsNull *fieldlist
```

NullAsValue

NullAsValue 문은 NULL을 값으로 변환해야 하는 필드를 지정합니다.

```
NullAsValue *fieldlist
```

Qualify

Qualify 문은 필드 이름의 한정을 설정하는 데 사용됩니다. 즉, 필드 이름 앞에 테이블 이름이 추가됩니다.

```
Qualify *fieldlist
```

Rem

rem 문은 스크립트에 설명 또는 주석을 삽입하거나 스크립트 문을 제거하지 않고 일시적으로 비활성화할 때 사용됩니다.

```
Rem string
```

Rename Field
이 스크립트 함수는 기존 Qlik Sense 필드를 하나 이상 로드한 후 해당 필드의 이름을 바꿉니다.

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

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

Rename Table
이 스크립트 함수는 기존 Qlik Sense 내부 테이블을 하나 이상 로드한 후 해당 테이블의 이름을 바꿉니다.

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

Section
section 문을 사용하면 후속 **LOAD** 및 **SELECT** 문을 데이터 또는 액세스 권한의 정의로 고려할지 정의할 수 있습니다.

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

Select
ODBC 데이터 소스 또는 OLE DB 제공자의 필드는 표준 SQL **SELECT** 문을 통해 선택됩니다. 그러나 **SELECT** 문의 허용 여부는 사용되는 ODBC 드라이버 또는 OLE DB 제공자에 의해 좌우됩니다.

```
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
set 문은 스크립트 변수를 정의하는 데 사용됩니다. 문자열, 경로, 드라이브 등을 대체하는 데 사용할 수 있습니다.

```
Set variablename=string
```

Sleep
sleep 문은 지정된 시간 동안 스크립트 실행을 일시 중지합니다.

```
Sleep n
```

SQL
SQL 문을 사용하면 ODBC 또는 OLE DB 연결을 통해 임의의 SQL 명령을 전송할 수 있습니다.

```
SQL sql_command
```

SQLColumns
sqlcolumns 문은 **connect**가 설정된 ODBC 또는 OLE DB 데이터 소스의 열을 나타내는 필드 집합을 반환합니다.

```
SQLColumns
```

SQLTables
sqltables 문은 **connect**가 설정된 ODBC 또는 OLE DB 데이터 소스의 테이블을 나타내는 필드 집합을 반환합니다.

```
SQLTables
```

SQLTypes
sqltypes 문은 **connect**가 설정된 ODBC 또는 OLE DB 데이터 소스의 유형을 나타내는 필드 집합을 반환합니다.

```
SQLTypes
```

Star
데이터베이스의 모든 필드 값 집합을 표시하기 위해 사용되는 문자열은 **star** 문을 통해 설정할 수 있습니다. 후속 **LOAD** 문 및 **SELECT** 문에 영향을 줍니다.

```
Star is [ string ]
```

Store
Store 문은 QVD, Parquet, CSV 또는 TXT 파일을 만듭니다.

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

Tag
이 스크립트 문은 하나 이상의 필드 또는 테이블에 태그를 할당하는 방법을 제공합니다. 앱에 없는 필드 또는 테이블에 태그를 지정하려고 하는 경우 태그는 무시됩니다. 필드 또는 태그 이름의 충돌이 발견되는 경우 마지막 값이 사용됩니다.

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

Trace
trace 문이 사용되면 문자열을 **스크립트 실행 진행률** 창과 스크립트 로그 파일에 기록합니다. 이 기능은 디버깅 용도에 매우 유용합니다. **trace** 문보다 먼저 계산되는 변수의 $ 확장을 사용하면 메시지를 사용자 지정할 수 있습니다.

```
Trace string
```

Unmap
Unmap 문은 이후에 로드된 필드를 위해 이전의 **Map ... Using** 문을 사용하여 지정된 필드 값 매핑을 비활성화합니다.

```
Unmap *fieldlist
```

Unqualify

Unqualify 문은 이전에 **Qualify** 문으로 활성화된 필드 이름의 정규화를 비활성화하는 데 사용됩니다.

```
Unqualify *fieldlist
```

Untag

이 스크립트 문은 필드 또는 테이블에서 태그를 제거하는 방법을 제공합니다. 앱에 없는 필드 또는 테이블에서 태그를 제거하려고 하는 경우, 태그 제거는 무시됩니다.

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

Alias

alias 문은 이어지는 스크립트 내에서 발생할 때마다 필드 이름을 변경하는 데 사용할 별칭을 설정하는 데 사용됩니다.

구문:

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

인수:

인수

인수	설명
fieldname	소스 데이터의 필드 이름
aliasname	대신 사용할 별칭 이름

예 및 결과:

예	결과
`Alias ID_N as NameID;`	
`Alias A as Name, B as Number, C as Date;`	이 문 전체에 정의된 이름 변경 내용이 모든 후속 **SELECT** 및 **LOAD** 문에 사용됩니다. 스크립트에서 이후의 아무 위치에서나 새 **alias** 문을 통해 필드 이름의 새로운 별칭을 정의할 수 있습니다.

AutoNumber

이 문은 스크립트 실행 중에 발견된 필드의 평가된 각 고유 값에 해당하는 고유 정수 값을 생성합니다.

LOAD 문에서도 *autonumber (page 553)* 함수를 사용할 수 있지만 최적화된 로드를 사용하려면 일부 제한이 있습니다. 먼저 **QVD** 파일에서 데이터를 로드하고 **AutoNumber** 문을 사용하여 값을 기호 키로 변환하여 최적화된 로드를 만들 수 있습니다.

구문:

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

인수:

인수

인수	설명
*fieldlist	값을 고유한 정수 값으로 바꿔야 하는 쉼표로 구분된 필드 목록입니다. 필드 이름에 ?나 *와 같은 와일드카드 문자를 사용하여 일치하는 이름을 가진 모든 필드를 포함할 수 있습니다. 또한 *를 사용하여 모든 필드를 포함할 수 있습니다. 와일드카드를 사용할 때는 필드 이름에 인용 부호를 사용해야 합니다.
namespace	**Using** 네임스페이스는 선택 사항입니다. 서로 다른 필드의 동일한 값이 같은 키를 공유하는 네임스페이스를 만들려는 경우 이 옵션을 사용할 수 있습니다. 이 옵션을 사용하지 않으면 모든 필드는 별도의 키 인덱스를 갖게 됩니다.

제한 사항:

스크립트에 여러 **LOAD** 문이 있으면 마지막 **LOAD** 문 뒤에 **AutoNumber** 문을 배치해야 합니다.

예 - AutoNumber를 사용하는 스크립트

스크립트 예

이 예에서 데이터는 **AutoNumber** 문 없이 먼저 로드됩니다. 그런 다음 효과를 표시하기 위해 **AutoNumber** 문이 추가됩니다.

예에 사용된 데이터

데이터 로드 편집기에서 다음 데이터를 인라인 로드로 로드하여 아래 스크립트 예를 만듭니다. 지금은 **AutoNumber** 문을 주석 처리된 상태로 두십시오.

```
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;
```

시각화 만들기

Qlik Sense 시트에 두 개의 테이블 시각화를 만듭니다. **KeyToOtherTable**, **Region**, **Year**, **Month** 및 **Sales**를 첫 번째 테이블에 차원으로 추가합니다. **KeyToOtherTable**, **Region**, **Year**, **Month** 및 **Budget**을 두 번째 테이블에 차원으로 추가합니다.

결과

RegionSales 테이블

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

Budget 테이블

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

설명

이 예에서는 두 테이블을 연결하는 복합 필드 **KeyToOtherTable**을 보여 줍니다. **AutoNumber**는 사용되지 않습니다. **KeyToOtherTable** 값의 길이에 유의하십시오.

AutoNumber 문 추가

로드 스크립트에서 **AutoNumber** 문의 주석 처리를 제거합니다.

```
AutoNumber KeyToOtherTable;
```

결과

RegionSales 테이블

KeyToOtherTable	Region	Year	Month	Sales
1	North	2014	June	127
1	North	2014	May	245
2	North	2014	May	347
3	South	2013	May	221
4	South	2013	May	367
4	South	2014	June	645

Budget 테이블

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

설명

KeyToOtherTable 필드 값은 고유한 정수 값으로 대체되었으며 결과적으로 필드 값의 길이가 줄어들어 메모리가 절약되었습니다. 두 테이블의 키 필드는 **AutoNumber**의 영향을 받으며 테이블은 연결된 상태로 유지됩니다. 예는 데모용으로 간략하게 나와 있지만 많은 행이 포함된 테이블에서는 의미가 있습니다.

Binary

binary 문은 다른 Qlik Sense 앱 또는 QlikView 문서에서 섹션 액세스 데이터를 포함한 데이터를 로드하는 데 사용됩니다. 앱의 다른 요소(예: 시트, 스토리, 시각화, 마스터 항목 또는 변수)는 포함되지 않습니다.

스크립트에는 하나의 **binary** 문만 허용됩니다. **binary** 문은 스크립트의 첫 번째 문이어야 합니다. 또한 일반적으로 스크립트 시작 부분에 위치하는 SET 문 앞에 와야 합니다.

구문:

```
binary [path] filename
```

인수:

인수

인수	설명
path	폴더 데이터 연결에 대한 참조로 사용해야 할 파일의 경로입니다. 파일이 Qlik Sense 작업 디렉터리에 없는 경우 필요합니다. ***'lib://Table Files/'*** 레거시 스크립팅 모드에서는 다음 경로 형식도 지원됩니다. • 절대 경로 ***c:\data*** • 이 스크립트 행을 포함하는 앱의 상대 경로. ***data***
filename	파일 확장명 .qvw 또는 .qvf를 포함한 파일 이름입니다.

제한 사항:

앱 ID를 참조하여 동일한 Qlik Sense Enterprise 배포의 앱에서 데이터를 로드하려면 **binary**를 사용할 수 없습니다. *.qvf* 파일에서만 로드할 수 있습니다.

예

문자열	설명
`Binary lib://DataFolder/customer.qvw;`	이 예에서 파일은 **폴더** 데이터 연결 위치에 있어야 합니다. 이 위치는 관리자가 Qlik Sense 서버에 만든 폴더일 수 있습니다. 데이터 로드 편집기에서 **새 연결 만들기**를 클릭한 다음 **파일 위치**에서 **폴더**를 선택합니다.
`Binary customer.qvf;`	이 예에서는 파일이 Qlik Sense 작업 디렉터리에 있어야 합니다.
`Binary c:\qv\customer.qvw;`	절대 파일 경로를 사용하는 이 예는 레거시 스크립팅 모드에서만 작동합니다.

Comment field

데이터베이스와 스프레드시트의 필드 주석(메타데이터)을 표시하는 방법을 제공합니다. 앱에 없는 필드 이름은 무시됩니다. 필드 이름이 여러 번 발견되는 경우 마지막 값이 사용됩니다.

구문:

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

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

맵 테이블에는 두 개의 열이 있어야 하며, 첫째 열은 필드 이름을, 둘째 열은 주석을 포함해야 합니다.

인수:

인수

인수	설명
**fieldlist*	주석을 추가할 필드의 쉼표로 구분된 목록입니다. 필드 목록에 *를 사용하면 모든 필드를 지정할 수 있습니다. 필드 이름에는 와일드카드 문자 * 및 ?가 허용됩니다. 와일드카드를 사용할 경우 필드 이름을 따옴표로 묶어야 합니다.
mapname	이전에 mapping **LOAD** 또는 mapping **SELECT** 문으로 읽은 매핑 테이블의 이름입니다.
fieldname	주석을 추가할 필드의 이름입니다.
comment	필드에 추가할 주석입니다.

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

데이터베이스나 스프레드시트의 테이블 주석(메타데이터)을 표시하는 방법을 제공합니다.

앱에 없는 테이블 이름은 무시됩니다. 테이블 이름이 여러 번 발견되는 경우 마지막 값이 사용됩니다. 키워드를 사용하여 데이터 소스에서 주석을 읽을 수 있습니다.

구문:

```
comment [tables] tablelist using mapname
```

```
comment [table] tablename with comment
```

인수:

인수

인수	설명
tablelist	(table{,table})
mapname	이전에 mapping **LOAD** 또는 mapping **SELECT** 문으로 읽은 매핑 테이블의 이름입니다.
tablename	주석을 추가할 테이블의 이름입니다.
comment	테이블에 추가할 주석입니다.

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

CONNECT 문은 OLE DB/ODBC 인터페이스를 통해 일반 데이터베이스에 대한 Qlik Sense 액세스를 정의하는 데 사용됩니다. ODBC의 경우, 먼저 ODBC 관리자를 사용하여 데이터 소스를 지정해야 합니다.

이 기능은 Qlik Sense SaaS에서 사용할 수 없습니다.

이 구문은 표준 모드의 폴더 데이터 연결만 지원합니다.

구문:

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

인수:

인수

인수	설명
connect-string	connect-string ::= datasourcename { ; conn-spec-item } 연결 문자열은 데이터 소스 이름과 하나 이상의 연결 사양 항목을 포함하는 선택적인 목록으로 구성됩니다. 데이터 소스 이름에 공백이 포함되어 있거나 연결 사양 항목이 나열된 경우 연결 문자열을 인용 부호로 묶어야 합니다. **datasourcename**은 정의된 ODBC 데이터 소스이거나 OLE DB 공급자를 정의하는 문자열이어야 합니다. conn-spec-item ::=**DBQ=**database_specifier |**DriverID=**driver_specifier |**UID=**userid |**PWD=**password 사용 가능한 연결 사양 항목은 데이터베이스에 따라 다를 수 있습니다. 일부 데이터베이스의 경우 위의 항목 외에 다른 항목도 사용할 수 있습니다. OLE DB의 경우 일부 연결 관련 항목이 선택 사항이 아닌 필수입니다.
connection	데이터 로드 편집기에 저장된 데이터 연결의 이름입니다.

CONNECT 앞에 **ODBC**가 있으면 ODBC 인터페이스가 사용되고, 그렇지 않으면 OLE DB가 사용됩니다.

LIB CONNECT TO를 사용하면 데이터 로드 편집기에서 만든 저장된 데이터 연결을 사용하여 데이터베이스에 연결합니다.

Example 1:

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

이 문을 통해 정의된 데이터 소스는 새로운 **CONNECT** 문이 만들어지기 전까지 후속 **Select (SQL)** 문에서 사용됩니다.

Example 2:

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

Connect32

이 문은 **CONNECT** 문과 동일한 방법으로 사용되며 64비트 시스템에서 32비트 ODBC/OLE DB 공급자를 사용하도록 지정합니다. 사용자 지정 연결에는 사용할 수 없습니다.

Connect64

이 문은 **CONNECT** 문과 동일한 방법으로 사용되며 64비트 공급자를 사용하도록 지정합니다. 사용자 지정 연결에는 사용할 수 없습니다.

Declare

Declare 문은 필드 또는 함수 간의 관계를 정의할 수 있는 필드 정의를 만드는 데 사용됩니다. 일련의 필드 정의는 차원으로 사용할 수 있는 파생된 필드를 자동으로 생성하는 데 사용할 수 있습니다. 예를 들어, 캘린더 정의를 만들 수 있으며 이를 사용하여 날짜 필드에서 연도, 월, 주, 일 등의 관련 차원을 생성할 수 있습니다.

Declare는 새 필드 정의를 설정하거나 기존 정의에 기반하여 필드 정의를 만들 때 사용할 수 있습니다.

새 필드 정의 설정

구문:

```
definition_name:

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

[Parameters parameter_list ]

Fields field_list
```

인수:

인수	설명
definition_name	필드 정의의 이름이며 콜론으로 끝납니다. *필드 정의에 대한 이름으로 autoCalendar를 사용하지 마십시오. 이 이름은 자동 생성 캘린더 템플릿용으로 예약되어 있습니다.* `Calendar:`
tag_list	필드 정의에서 파생된 필드에 적용할 태그의 쉼표로 구분된 목록입니다. 태그 적용은 선택 사항이지만, $date, $numeric 또는 $text 등과 같이 정렬 순서를 지정하는 데 사용되는 태그를 적용하지 않으면 파생 필드가 기본적으로 로드 순서에 따라 정렬됩니다. `'$date'Thank you for bringing this to our attention, and apologies for the inconvenience.`

인수	설명
parameter_list	쉼표로 구분된 매개 변수 목록입니다. 매개 변수는 name=value 형식으로 정의되며 필드 정의를 재사용할 때 재정의할 수 있는 시작 값이 할당됩니다. 선택 사항입니다. first_month_of_year = 1
field_list	필드 정의를 사용할 때 생성되는 쉼표로 구분된 필드 목록입니다. 필드는 <expression> **As** field_name **tagged** tag 형식으로 정의됩니다. $1을 사용하여, 파생 필드를 생성해야 하는 원본 데이터 필드를 참조합니다. Year($1) As Year tagged ('$numeric')

```
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')
;
```

이제 캘린더가 정의되고, 이를 **Derive** 절을 사용하여 로드된 데이터 필드(이 경우에는 OrderDate 및 ShippingDate)에 적용할 수 있습니다.

기존 필드 정의 재사용

구문:

```
<definition name>:

Declare [Field][s] Definition

Using <existing_definition>

[With <parameter_assignment> ]
```

인수:

인수	설명
definition_name	필드 정의의 이름이며 콜론으로 끝납니다. `MyCalendar:`
existing_definition	새 필드 정의를 만들 때 재사용할 필드 정의입니다. 새로운 필드 정의는, parameter_assignment를 사용하여 필드 표현식에 사용된 값을 변경하는 경우를 제외하고, 해당 정의의 기반으로 사용된 원본 정의와 동일한 방식으로 작동합니다. `Using Calendar`
parameter_assignment	쉼표로 구분된 매개 변수 할당 목록입니다. 매개 변수 할당은 `name=value` 형식으로 정의되며, 기본 필드 정의에 설정된 매개 변수 값을 재정의합니다. 선택 사항입니다. `first_month_of_year = 4`

이 예에서는 이전 예에서 만든 캘린더 정의를 재사용합니다. 이 사례의 경우 4월에 시작하는 회계년도를 사용하려고 합니다. 이 작업은 값 4를 first_month_of_year 매개 변수에 할당하여 수행되며, 정의된 DayNumberOfYear 필드에 영향을 미칠 수 있습니다.

이 예에서는 이전 예의 샘플 데이터와 필드 정의를 사용한다고 가정합니다.

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



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

데이터 스크립트를 다시 로드하면 OrderDate.MyCalendar.* 및 ShippingDate.MyCalendar.*라는 이름으로 생성된 필드를 시트 편집기에서 사용할 수 있습니다.

Derive

Derive 문은 **Declare** 문으로 만든 필드 정의를 바탕으로 파생된 필드를 생성하는 데 사용됩니다. 필드를 파생할 데이터 필드를 지정하거나 필드 태그를 바탕으로 명시적 또는 암시적으로 데이터 필드를 파생할 수 있습니다.

구문:

```
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
```

인수:

인수

인수	설명
definition	필드를 파생시킬 때 사용할 필드 정의의 이름입니다. **Calendar**
field_list	필드 정의에 기반하여 파생된 필드를 생성해야 하는 원본 데이터 필드의 쉼표로 구분된 목록입니다. 데이터 필드는 스크립트에서 이미 로드한 필드여야 합니다. **OrderDate, ShippingDate**
tag_list	태그의 쉼표로 구분된 목록입니다. 나열된 태그를 사용하는 모든 데이터 필드에 대해 파생된 필드가 생성됩니다. 태그 목록은 둥근 괄호로 묶어야 합니다. **('$date','$timestamp')**

- 특정 데이터 필드에 대한 필드를 파생합니다.
  이 경우에는 OrderDate 및 ShippingDate 필드를 지정합니다.
  `DERIVE FIELDS FROM FIELDS OrderDate,ShippingDate USING Calendar;`
- 특정 태그가 있는 모든 필드에 대해 필드를 파생합니다.
  이 경우에는 $date 태그가 있는 모든 필드에 대해 Calendar에 기반하여 필드를 파생시킵니다.
  `DERIVE FIELDS FROM EXPLICIT TAGS ('$date') USING Calendar;`
- 필드 정의 태그가 있는 모든 필드에 대해 필드를 파생합니다.
  이 경우에는 Calendar 필드 정의와 동일한 태그가 있는 모든 데이터 필드(이 경우, $date)에 대해 필드를 파생시킵니다.
  `DERIVE FIELDS FROM IMPLICIT TAG USING Calendar;`

Direct Query

DIRECT QUERY 문을 사용하면 Direct Discovery 함수를 사용하여 ODBC 또는 OLE DB 연결을 통해 테이블에 액세스할 수 있습니다.

구문:

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

DIMENSION, **MEASURE** 및 **DETAIL** 키워드는 어떤 순서로도 사용할 수 있습니다.

모든 **DIRECT QUERY** 문에는 **DIMENSION** 및 **FROM** 키워드 절이 필요합니다. **FROM** 키워드는 **DIMENSION** 키워드 뒤에 나와야 합니다.

DIMENSION 키워드 바로 다음에 지정되는 필드는 메모리에 로드되며, 인 메모리와 Direct Discovery 데이터 간 연결을 만드는 데 사용할 수 있습니다.

> ***DIRECT QUERY** 문에는 **DISTINCT** 또는 **GROUP BY** 절을 포함할 수 없습니다.*

MEASURE 키워드를 사용하면 Qlik Sense가 "메타 수준"에서 인식하는 필드를 정의할 수 있습니다. 데이터 로드 프로세스 중에는 측정값 필드의 실제 데이터가 데이터베이스 내에만 존재하며, 시각화에서 사용되는 차트 표현식을 통해 임시로 검색됩니다.

일반적으로 차원으로 사용될 불연속 값이 있는 필드는 **DIMENSION** 키워드로 로드해야 하며, 집계에만 사용될 숫자는 **MEASURE** 키워드로 선택해야 합니다.

DETAIL 필드는 사용자가 하위 수준 표시 테이블 상자에 표시하고자 하는 주석 필드와 같은 정보 또는 세부 사항을 제공합니다. **DETAIL** 필드는 차트 표현식에 사용할 수 없습니다.

기본적으로 **DIRECT QUERY** 문은 SQL을 지원하는 데이터 소스에 대해 데이터-소스 중립적입니다. 따라서 동일한 **DIRECT QUERY** 문을 변경하지 않고 서로 다른 SQL 데이터베이스에 사용할 수 있습니다. Direct Discovery는 필요에 따라 데이터베이스에 적합한 쿼리를 생성합니다.

네이티브 데이터-소스 구문은 사용자가 쿼리할 데이터베이스를 알고 있으며, SQL에 대한 데이터베이스 관련 확장 기능을 활용하고자 할 때 사용할 수 있습니다. 네이티브 데이터-소스 구문은 다음과 같이 지원됩니다.

- **DIMENSION** 및 **MEASURE** 절에서 필드 표현식으로
- **WHERE** 절의 내용으로

예:

```
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"')
```

> *다음 용어는 키워드로 사용되므로 따옴표 없이 열 또는 필드 이름으로 사용할 수 없습니다. and, as, detach, detail, dimension, distinct, from, in, is, like, measure, native, not, or, where*

인수:

인수	설명
fieldlist	필드 사양의 쉼표로 구분된 목록입니다. *fieldname {, fieldname}*. 필드 사양은 필드 이름일 수 있으며, 이 경우 데이터베이스 열 이름과 Qlik Sense 필드 이름에 동일한 이름이 사용됩니다. 또는 필드 사양이 "필드 별칭"이 될 수 있으며, 이 경우 데이터베이스 표현식 또는 열 이름에 Qlik Sense 필드 이름이 제공됩니다.
tablelist	데이터를 로드할 데이터베이스 내의 테이블 또는 보기 이름의 목록입니다. 일반적으로 이는 데이터베이스에서 수행된 JOIN을 포함하는 보기입니다.

where_clause	여기에는 데이터베이스 **WHERE** 절의 전체 구문이 정의되어 있지 않지만 함수 호출, 문자열에 대한 **LIKE** 연산자, **IS NULL** 및 **IS NOT NULL** 등의 사용을 포함하여 대부분의 SQL "관계형 표현식"이 허용되며, **BETWEEN**은 포함되지 않습니다.
	
	NOT은 특정 키워드에 대한 수정자와 달리 단항 연산자입니다.
	
	예:
	
	`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)`
	마지막 예는 다음과 같이 작성할 수 없습니다.
	
	`WHERE X NOT in (1,2,3)`

이 예에서는 Dim1, Dim2, Num1, Num2 및 Num3 필드가 있는 TableName 데이터베이스 테이블이 사용됩니다.Dim1 및 Dim2가 Qlik Sense 데이터 집합에 로드됩니다.

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

Dim1 및 Dim2는 차원으로 사용할 수 있습니다. Num1, Num2 및 Num3은 집계에 사용할 수 있습니다. Dim1 및 Dim2 또한 집계에 사용할 수 있습니다. Dim1 및 Dim2를 사용할 수 있는 집계의 유형은 데이터 유형에 따라 결정됩니다. 예를 들어 대부분의 경우 **DIMENSION** 필드에는 이름 또는 계좌 번호와 같은 문자열 데이터가 포함됩니다. 해당 필드는 합산할 수 없지만 계수될 수는 있습니다(`count(Dim1)`).

***DIRECT QUERY** 문은 스크립트 편집기에서 직접 작성합니다. **DIRECT QUERY** 문의 구조를 간소화하려면 데이터 연결에서 **SELECT** 문을 생성한 다음, 생성된 스크립트를 편집하여 **DIRECT QUERY** 문으로 바꿀 수 있습니다.*
*예를 들어 **SELECT** 문은 다음과 같습니다.*

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

*다음 **DIRECT QUERY** 문으로 변경될 수 있습니다.*

```
DIRECT QUERY
 DIMENSION
 SalesOrderID,
 RevisionNumber

MEASURE
 SubTotal,
 TaxAmt

DETAIL
OrderDate

FROM MyDB.Sales.SalesOrderHeader;
```

Direct Discovery 필드 목록

필드 목록은 필드 사양의 쉼표로 구분된 목록입니다(*fieldname {, fieldname}*). 필드 사양은 필드 이름일 수 있으며, 이 경우 데이터베이스 열 이름과 필드 이름에 동일한 이름이 사용됩니다. 또는 필드 사양이 필드 별칭이 될 수 있으며, 이 경우 데이터베이스 표현식 또는 열 이름에 Qlik Sense 필드 이름이 제공됩니다.

필드 이름은 단순한 이름 또는 인용된 이름일 수 있습니다. 단순한 이름은 알파벳 유니코드 문자로 시작되고 알파벳 또는 순자 문자나 밑줄의 조합이 뒤에 옵니다. 인용된 이름은 큰따옴표로 시작되며 임의 순서의 문자를 포함합니다. 인용된 이름에 큰따옴표가 있을 경우 해당 인용 부호는 2개의 인접한 큰따옴표를 사용하여 표현됩니다.

Qlik Sense 필드 이름은 대/소문자가 구분됩니다. 데이터베이스 필드 이름은 데이터베이스에 따라 대/소문자가 구분되거나 구분되지 않을 수 있습니다. Direct Discovery 쿼리는 모든 필드 식별자와 별칭의 대/소문자를 유지합니다. 다음 예에서는 데이터베이스 열 "STATEID"의 데이터를 저장하기 위해 내부적으로 "MyState"라는 별칭을 사용합니다.

```
DIRECT QUERY Dimension STATEID as MyState Measure AMOUNT from SALES_TABLE;
```

따라서 별칭을 사용한 **SQL Select** 문의 결과와 다릅니다. 별칭이 명시적으로 인용되지 않은 경우 결과에 대상 데이터베이스에서 반환한 열의 기본 대/소문자가 포함됩니다. 다음 예에서는 별칭이 대/소문자 혼합으로 지정되더라도 Oracle 데이터베이스에 대한 **SQL Select** 문이 Qlik Sense 내부 별칭처럼 모두 대문자인 "MYSTATE,"를 만듭니다. **SQL Select** 문은 데이터베이스에서 반환된 열 이름을 사용하며, Oracle의 경우 모두 대문자로 된 이름을 사용합니다.

```
SQL Select STATEID as MyState, STATENAME from STATE_TABLE;
```

이러한 동작을 방지하려면 LOAD 문을 사용하여 별칭을 지정하십시오.

```
Load STATEID as MyState, STATENAME;
SQL Select STATEID, STATEMENT from STATE_TABLE;
```

이 예에서는 "STATEID" 열이 Qlik Sense에 의해 내부적으로 "MyState"로 저장됩니다.

대부분의 데이터베이스 스칼라 표현식은 필드 사양으로 허용됩니다. 함수 호출 또한 필드 사양에 사용할 수 있습니다. 표현식에는 작은따옴표 안에 포함된 부울, 숫자 또는 문자열 상수가 포함될 수 있습니다(포함된 작은 따옴표는 인접한 작은 따옴표를 사용하여 표시됩니다).

```
DIRECT QUERY
   DIMENSION
      SalesOrderID, RevisionNumber
   MEASURE
      SubTotal AS "Sub Total"
FROM AdventureWorks.Sales.SalesOrderHeader;

DIRECT QUERY
   DIMENSION
      "SalesOrderID" AS "Sales Order ID"
   MEASURE
      SubTotal,TaxAmt,(SubTotal-TaxAmt) AS "Net Total"
FROM AdventureWorks.Sales.SalesOrderHeader;

DIRECT QUERY
   DIMENSION
      (2*Radius*3.14159) AS Circumference,
      Molecules/6.02e23 AS Moles
```

```
        MEASURE

           Num1 AS numA

     FROM TableName;
```

```
DIRECT QUERY
   DIMENSION
      concat(region, 'code') AS region_code
   MEASURE
      Num1 AS NumA
FROM TableName;
```

Direct Discovery는 **LOAD** 문에서의 집계 사용을 지원하지 않습니다. 집계를 사용하면 결과를 예측할 수 없게 됩니다. 다음과 같이 **LOAD** 문을 사용해서는 안 됩니다.

```
DIRECT QUERY DIMENSION stateid, SUM(amount*7) AS MultiFirst MEASURE amount FROM sales_table;
```

LOAD 문에 **SUM**을 사용해서는 안 됩니다.

또한 Direct Discovery는 **Direct Query** 문에 Qlik Sense 함수를 사용하는 것도 지원하지 않습니다. 예를 들어 **DIMENSION** 필드의 사양이 다음과 같은 경우 시각화에서 "Mth" 필드를 차원으로 사용하면 오류가 발생합니다.

```
month(ModifiedDate) as Mth
```

Directory

Directory 문은 새 **Directory** 문이 만들어질 때까지 후속 **LOAD** 문에서 데이터 파일을 탐색할 디렉터리를 정의합니다.

구문:

```
Directory[path]
```

Directory 문을 **path** 없이 실행하거나 생략하는 경우 Qlik Sense는 Qlik Sense 작업 디렉터리에서 검색합니다.

인수:

인수

인수	설명
path	data 파일의 경로로 해석할 수 있는 텍스트입니다. 경로는 파일에 대한 경로이며, 다음 중 하나입니다. • 절대 경로 ***c:\data*** • Qlik Sense 앱 작업 디렉터리에 대한 상대 경로 ***data*** • 인터넷 또는 인트라넷상의 위치를 가리키는 URL 주소(HTTP 또는 FTP) ***http://www.qlik.com***

```
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

Disconnect 문은 현재 ODBC/OLE DB/사용자 지정 연결을 종료합니다. 이 문은 옵션입니다.

구문:

```
Disconnect
```

새로운 **connect** 문이 실행되거나 스크립트 실행이 완료되면 연결이 자동으로 종료됩니다.

```
Disconnect;
```

Drop

Drop 스크립트 키워드는 테이블 또는 필드를 데이터베이스에서 삭제하는 데 사용할 수 있습니다.

Drop field

스크립트 실행 도중 언제든 **drop field** 문을 사용하여 데이터 모델, 즉 메모리에서 하나 또는 여러 개의 Qlik Sense 필드를 삭제할 수 있습니다. 테이블의 "distinct" 속성은 **drop field** 문 다음에 제거됩니다.

> ***drop field** 및 **drop fields**는 모두 허용되는 형식이며 결과에 차이가 없습니다. 테이블을 지정하지 않으면 필드가 발생되는 모든 테이블에서 삭제됩니다.*

구문:

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

```
Drop field A;
Drop fields A,B;
Drop field A from X;
Drop fields A,B from X,Y;
```

Drop table

스크립트 실행 도중 언제든 **drop table** 문을 사용하여 데이터 모델, 즉 메모리에서 하나 또는 여러 개의 Qlik Sense 내부 테이블을 삭제할 수 있습니다.

구문:

```
drop table  tablename {, tablename2 ...}
drop tables tablename {, tablename2 ...}
```

> ***drop table** 및 **drop tables** 형식을 둘 다 사용할 수 있습니다.*

이 작업의 결과로 다음 항목이 손실됩니다.

- 실제 테이블
- 나머지 테이블에 속하지 않는 모든 필드
- 삭제된 테이블에서 배타적으로 가져온 나머지 필드의 필드 값

예 및 결과:

예	결과
`drop table Orders, Salesmen, T456a;`	메모리에서 3개의 테이블을 삭제하는 줄입니다.

예	결과
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;	테이블 *Tab2*가 생성된 후 테이블 *Tab1*이 삭제됩니다.

Drop table

스크립트 실행 도중 언제든 **drop table** 문을 사용하여 데이터 모델, 즉 메모리에서 하나 또는 여러 개의 Qlik Sense 내부 테이블을 삭제할 수 있습니다.

구문:

```
drop table  tablename {, tablename2 ...}
drop tables tablename {, tablename2 ...}
```

***drop table** 및 **drop tables** 형식을 둘 다 사용할 수 있습니다.*

이 작업의 결과로 다음 항목이 손실됩니다.

- 실제 테이블
- 나머지 테이블에 속하지 않는 모든 필드
- 삭제된 테이블에서 배타적으로 가져온 나머지 필드의 필드 값

예 및 결과:

예	결과
drop table Orders, Salesmen, T456a;	메모리에서 3개의 테이블을 삭제하는 줄입니다.
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;	테이블 *Tab2*가 생성된 후 테이블 *Tab1*이 삭제됩니다.

Execute

Execute 문은 Qlik Sense가 데이터를 로드하는 동안 다른 프로그램을 실행하는 데 사용됩니다. 예를 들어, 필요한 규칙을 만드는 데 사용됩니다.

이 기능은 Qlik Sense SaaS에서 사용할 수 없습니다.

표준 모드에서는 이 구문이 지원되지 않습니다.

구문:

```
execute commandline
```

인수:

인수

인수	설명
commandline	운영 체제에서 명령줄로 해석할 수 있는 텍스트입니다. 절대 파일 경로 또는 lib:// 폴더 경로를 참조할 수 있습니다.

Execute를 사용하려면 다음 조건을 충족해야 합니다.

- 레거시 모드(Qlik Sense 및 Qlik Sense Desktop에서 사용 가능)로 실행해야 합니다.
- *Settings.ini*(Qlik Sense에서 사용 가능)에서 OverrideScriptSecurity를 1로 설정해야 합니다. *Settings.ini*는 *C:\ProgramData\Qlik\Sense\Engine*에 있으며 일반적으로 빈 파일입니다.

***Execute**를 활성화하도록 OverrideScriptSecurity를 설정한 경우, 모든 사용자가 서버의 파일을 실행할 수 있습니다. 예를 들어, 사용자가 앱에 실행 가능한 파일을 첨부하고 데이터 로드 스크립트에서 해당 파일을 실행할 수 있습니다.*

다음과 같이 하십시오.

1. *Settings.ini*의 복사본을 만들고 텍스트 편집기에서 엽니다.
2. 파일이 첫 줄에 *[Settings 7]*이 포함되어 있는지 확인합니다.
3. 새 줄을 삽입하고 *OverrideScriptSecurity=1*을 입력합니다.
4. 파일의 끝 부분에 빈 줄을 삽입합니다.
5. 파일을 저장합니다.
6. 편집된 파일로 *Settings.ini*를 대체합니다.
7. Qlik Sense Engine Service (QES)을 다시 시작합니다.

Qlik Sense가 서비스로 실행 중이면 일부 명령이 예상대로 작동하지 않을 수 있습니다.

```
Execute C:\Program Files\Office12\Excel.exe;
Execute lib://win\notepad.exe // win is a folder connection referring to c:\windows
```

Field/Fields

Field 및 **Fields** 스크립트 키워드는 **Declare**, **Derive**, **Drop**, **Comment**, **Rename** 및 **Tag/Untag** 문에서 사용됩니다.

FlushLog

FlushLog 문을 사용하면 Qlik Sense에서 스크립트 로그 파일에 스크립트 버퍼의 내용을 기록할 수 있습니다.

구문:

```
FlushLog
```

버퍼의 내용이 로그 파일에 기록됩니다. 그렇지 않으면 실패한 스크립트 실행 시 손실되었을 수 있는 데이터를 수신하게 되므로 이 명령은 디버깅 용도에 매우 유용할 수 있습니다.

```
FlushLog;
```

Force

force 문은 Qlik Sense가 후속 **LOAD** 및 **SELECT** 문의 필드 이름 및 필드 값을 대문자 또는 소문자로만 해석하거나 첫 글자를 항상 대문자로 해석하거나 원래대로(혼합) 해석하도록 지정합니다. 이 문을 사용하여 다른 규칙이 적용된 테이블의 필드 값을 연결할 수 있습니다.

구문:

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

아무것도 지정하지 않으면 force case mixed가 가정됩니다. force 문은 새로운 force 문이 나올 때까지 적용됩니다.

force 문은 액세스 섹션에 영향을 미치지 않으며 로드된 모든 필드 값은 대/소문자를 구분하지 않습니다.

예 및 결과

예	결과

이 예에서는 첫 문자를 대문자로 표시하는 방법을 보여줍니다.
`FORCE Capitalization;`
`Capitalization:`
`LOAD * Inline [`
`ab`
`Cd`
`eF`
`GH`
`];`
	Capitalization 테이블에는 다음 값이 포함되어 있습니다.
	`Ab`
	`Cd`
	`Ef`
	`Gh`
	모든 값의 첫 문자가 대문자로 표시됩니다.

이 예에서는 강제로 대문자로 표시하는 방법을 보여줍니다.
`FORCE Case Upper;`
`CaseUpper:`
`LOAD * Inline [`
`ab`
`Cd`
`eF`
`GH`
`];`
	CaseUpper 테이블에는 다음 값이 포함되어 있습니다.
	`AB`
	`CD`
	`EF`
	`GH`
	모든 값이 대문자입니다.

예	결과

이 예에서는 강제로 소문자로 표시하는 방법을 보여줍니다.
`FORCE Case Lower;`
`CaseLower:`
`LOAD * Inline [`
`ab`
`Cd`
`eF`
`GH`
`];`
	CaseLower 테이블에는 다음 값이 포함되어 있습니다.
	`ab`
	`cd`
	`ef`
	`gh`
	모든 값이 소문자입니다.

이 예에서는 대소문자를 혼합하여 표시하는 방법을 보여줍니다.
`FORCE Case Mixed;`
`CaseMixed:`
`LOAD * Inline [`
`ab`
`Cd`
`eF`
`GH`
`];`
	CaseMixed 테이블에는 다음 값이 포함되어 있습니다.
	`ab`
	`Cd`
	`eF`
	`GH`
	모든 값은 스크립트에 나오는 대로 표시됩니다.

관련 항목:

From

From 스크립트 키워드는 **Load** 문에서 파일을 참조하는 데 사용되며 **Select** 문에서는 데이터베이스 테이블 또는 뷰를 참조하는 데 사용됩니다.

Load

LOAD 문은 파일, 스크립트에 정의된 데이터, 이전에 로드한 테이블, 웹 페이지, 이후 **SELECT** 문의 결과에서 필드를 로드하거나 자동으로 데이터를 생성하여 필드를 로드합니다. 분석 연결에서 데이터를 로드할 수도 있습니다.

구문:

```
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 ]
```

인수:

인수

인수	설명
distinct	고유한 레코드만 로드하려는 경우 **distinct**를 조건자로 사용할 수 있습니다. 중복 레코드가 있는 경우 첫 번째 인스턴스가 로드됩니다. 선행 LOAD를 사용하는 경우 **distinct**는 대상 테이블에만 영향을 주므로 첫 번째 Load 문에 **distinct**를 배치해야 합니다.

인수	설명
fieldlist	*fieldlist ::= (* | field{, * | field })* 로드할 필드의 목록입니다. 필드 목록에 *를 사용하면 테이블의 모든 필드를 지정할 수 있습니다. *field ::= (fieldref | expression) [***as** *aliasname]* 필드 정의는 리터럴, 기존 필드에 대한 참조 또는 표현식을 항상 포함해야 합니다. *fieldref ::= (fieldname |@fieldnumber |@startpos:endpos [**I** | **U** | **R** | **B** | **T**])* *fieldname*은 테이블의 필드 이름과 동일한 텍스트입니다. 필드 이름에 공백 등이 포함된 경우 곧은 큰따옴표 또는 대괄호로 묶어야 합니다. 필드 이름을 명시적으로 사용할 수 없는 경우도 있습니다. 이 경우 다른 표기법이 사용됩니다. *@fieldnumber*는 구분된 테이블 파일의 필드 번호를 나타냅니다. 이 숫자는 "@"가 앞에 오는 양의 정수여야 합니다. 숫자는 항상 1부터 시작하며 필드 개수까지 지정할 수 있습니다. *@startpos*:*endpos*는 고정 길이 레코드를 가진 파일 내 필드의 시작과 끝 위치를 나타냅니다. 위치는 모두 양의 정수여야 합니다. 이 두 숫자는 "@"가 앞에 와야 하며 콜론으로 구분되어야 합니다. 숫자는 항상 1부터 시작하며 위치 개수까지 지정할 수 있습니다. 마지막 필드의 **n**은 종료 위치로 사용됩니다. • *@startpos*:*endpos*의 바로 뒤에 문자 **I** 또는 **U**가 오는 경우 로드된 바이트는 부호가 있는(**I**) 이진수 또는 부호가 없는(**U**) 정수(Intel 바이트 순서)로 해석됩니다. 로드된 위치의 수는 1, 2 또는 4여야 합니다. • *@startpos*:*endpos*의 바로 뒤에 문자 **R**이 오는 경우, 로드된 바이트는 이진 실수(IEEE 32비트 또는 64비트 부동 소수점)로 해석됩니다. 로드된 위치의 수는 4 또는 8이어야 합니다. • *@startpos*:*endpos*의 바로 뒤에 문자 **B**가 오는 경우, 로드된 바이트는 COMP-3 표준에 따라 BCD (Binary Coded Decimal) 숫자로 해석됩니다. 임의의 바이트 수가 지정될 수 있습니다. *expression*은 같은 테이블 내의 하나 또는 여러 다른 필드에 따라 숫자 함수 또는 문자열 함수가 될 수 있습니다. 자세한 내용은 표현식의 구문을 참조하십시오. **as**는 필드에 새 이름을 할당하는 데 사용됩니다.

인수	설명
from	**from**은 폴더 또는 웹 파일 데이터 연결을 사용하여 파일에서 데이터를 로드해야 하는 경우에 사용됩니다. *file ::= [path] filename* ***'lib://Table Files/'*** 경로를 생략한 경우 Qlik Sense는 **Directory** 문으로 지정된 디렉터리에서 파일을 검색합니다. **Directory** 문이 없으면 Qlik Sense가 작업 디렉터리 *C:\Users\{user}\Documents\Qlik\Sense\Apps*에서 검색합니다. *Qlik Sense 서버 설치에서 작업 디렉터리는 Qlik Sense Repository Service에서 지정되며, 기본적으로 C:\ProgramData\Qlik\Sense\Apps 입니다.* *filename*에는 표준 DOS 와일드카드 문자(* 및 ?)가 포함될 수 있습니다. 이는 지정된 디렉터리 내에서 일치하는 파일이 모두 로드되도록 합니다. *format-spec ::= (fspec-item { , fspec-item })* 서식 사양은 괄호 안에 포함된 여러 서식 사양 항목 목록으로 구성됩니다. **레거시 스크립팅 모드** 레거시 스크립팅 모드에서는 다음 경로 형식도 지원됩니다. • 절대 경로 ***c:\data*** • Qlik Sense 앱 작업 디렉터리에 대한 상대 경로. ***data*** • 인터넷 또는 인트라넷상의 위치를 가리키는 URL 주소(HTTP 또는 FTP) ***http://www.qlik.com*** •

인수	설명
from_field	**from_field**는 이전에 로드한 필드에서 데이터를 로드해야 하는 경우에 사용됩니다. *fieldassource::=***(***tablename, fieldname***)*** 이 필드는 이전에 로드한 *tablename* 및 *fieldname*의 이름입니다. *format-spec ::= (fspec-item {, fspec-item })* 서식 사양은 괄호 안에 포함된 여러 서식 사양 항목 목록으로 구성됩니다. 자세한 내용은 *서식 사양 항목 (page 162)*를 참조하십시오. ***from_field****은 테이블에서 필드를 구분할 때 목록 구분 기호로 쉼표만 지원합니다.*
inline	**inline**은 데이터를 스크립트 내에 입력해야 하고 파일에서 로드되지 않도록 해야 하는 경우에 사용됩니다. *data ::= [text]* **inline** 절을 통해 입력된 데이터는 큰따옴표 또는 대괄호로 묶어야 합니다. 그 사이에 입력되는 텍스트는 파일의 내용과 동일한 방식으로 해석됩니다. 그러므로 텍스트 파일에 새 줄을 삽입하는 경우 **inline** 절의 텍스트에도 스크립트를 입력하면서 Enter 키를 누르는 등의 방법으로 동일하게 새 줄을 삽입해야 합니다. 첫 번째 줄에 열 수를 정의합니다. *format-spec ::= (fspec-item {, fspec-item })* 서식 사양은 괄호 안에 포함된 여러 서식 사양 항목 목록으로 구성됩니다. 자세한 내용은 *서식 사양 항목 (page 162)*를 참조하십시오.
resident	**resident**는 이전에 로드한 테이블에서 데이터를 로드해야 하는 경우에 사용됩니다. *table label*은 원래 테이블을 만든 **LOAD** 또는 **SELECT** 문 앞에 오는 레이블입니다. 이 레이블은 콜론으로 끝나야 합니다.
autogenerate	**autogenerate**는 Qlik Sense에서 자동으로 데이터를 생성해야 하는 경우에 사용됩니다. *size ::= number* *Number*는 생성할 레코드의 수를 나타내는 정수입니다. **Peek** 함수를 사용하여 이전에 로드한 테이블에서 단일 필드 값을 참조하지 않는 한, 필드 목록에는 외부 데이터 소스 또는 이전에 로드한 테이블로부터 데이터를 요구하는 표현식을 포함하지 않아야 합니다.

인수	설명
extension	분석 연결에서 데이터를 로드할 수 있습니다. 서버 측 확장(SSE) 플러그인에 정의된 함수를 호출하거나 스크립트를 평가하려면 **extension** 절을 사용해야 합니다. 단일 테이블을 SSE 플러그인에 보내면 단일 데이터 테이블이 반환됩니다. 플러그인이 반환되는 필드의 이름을 지정하지 않으면 필드의 이름이 Field1, Field2로 지정되는 식으로 계속됩니다. `Extension pluginname.functionname( tabledescription );` • SSE 플러그인의 함수를 사용하여 데이터 로드 *tabledescription ::= (table { ,tablefield})* 테이블 필드를 명시하지 않으면 필드가 로드 순서로 사용됩니다. • SSE 플러그인에서 스크립트를 평가하여 데이터 로드 *tabledescription ::= (script, table { ,tablefield})* **테이블 필드 정의에서 데이터 유형 처리** 데이터 유형은 분석 연결에서 자동으로 감지됩니다. 데이터에 숫자 값과 적어도 하나의 NULL이 아닌 텍스트 문자열이 있으면 이 필드가 텍스트로 간주됩니다. 다른 경우에는 숫자로 간주됩니다. **String()** 또는 **Mixed()**로 필드 이름을 래핑하여 데이터 유형을 강제로 설정할 수 있습니다. • **String()**은 필드를 텍스트로 강제 설정합니다. 필드가 숫자인 경우 이중 값의 텍스트 부분이 추출되고 변환은 수행되지 않습니다. • **Mixed()**는 필드를 이중으로 강제 설정합니다. **String()** 또는 **Mixed()**는 **extension** 테이블 필드 정의 외부에서 사용할 수 없으며 테이블 필드 정의에서 다른 Qlik Sense 함수를 사용할 수 없습니다. **분석 연결 추가 정보** 분석 연결을 사용하려면 먼저 구성해야 합니다.
where	**where**는 레코드를 선택에 포함할지를 나타내는 데 사용하는 절입니다. *criterion*이 True인 경우 선택 내용이 포함됩니다. *criterion*은 논리 표현식입니다.
while	**while**은 레코드를 반복적으로 읽어야 할지 여부를 지정하는 데 사용되는 절입니다. *criterion*이 True이면 동일한 레코드를 읽습니다. **while** 절을 유용하게 사용하려면 일반적으로 **IterNo( )** 함수를 포함해야 합니다. *criterion*은 논리 표현식입니다.

인수	설명
group by	**group by**는 데이터를 집계(그룹화)해야 하는 필드를 정의하는 데 사용되는 절입니다. 집계 필드는 로드한 표현식에 어떤 방식으로든 포함되어야 합니다. 집계 필드 외의 필드는 로드한 표현식의 집계 함수 외부에서 사용할 수 없습니다. *groupbyfieldlist ::= (fieldname { ,fieldname })*
order by	**order by**는 상주 테이블의 레코드가 **load** 문에 의해 처리되기 전에 정렬하는 데 사용되는 절입니다. 상주 테이블은 하나 이상의 필드를 기준으로 오름차순 또는 내림차순으로 정렬할 수 있습니다. 기본적으로 숫자 값을 기준으로 정렬되며, 2차적으로 국가별 정렬 순서에 따라 정렬됩니다. 이 절은 데이터 소스가 상주 테이블인 경우에만 사용할 수 있습니다. 정렬 필드는 상주 테이블의 정렬 기준 필드를 지정합니다. 이 필드는 상주 테이블 내의 이름 또는 번호(첫 번째 필드는 1번)로 지정할 수 있습니다. orderbyfieldlist ::= fieldname [sortorder] { , fieldname [sortorder] } *sortorder*는 *asc*(오름차순) 또는 *desc*(내림차순)입니다. *sortorder*를 지정하지 않으면 *asc*가 사용됩니다. *fieldname*, *path*, *filename* 및 *aliasname*은 각 이름의 의미를 나타내는 텍스트 문자열입니다. 소스 테이블의 모든 필드를 *fieldname*으로 사용할 수 있습니다. 그러나 as 절(*aliasname*)을 통해 만든 필드는 해당되지 않으며 같은 **load** 문에서 사용할 수 없습니다.

from, **inline**, **resident**, **from_field, extension** 또는 **autogenerate** 절을 통해 지정된 데이터 소스가 없는 경우 데이터는 바로 다음에 나오는 **SELECT** 또는 **LOAD** 문의 결과에서 로드됩니다. 다음에 나오는 문에는 접두사가 없어야 합니다.

다른 파일 형식 로드
기본 옵션으로 구분된 데이터 파일 로드:

```
LOAD * from data1.csv;
```

라이브러리 연결에서 구분된 데이터 파일 로드(DataFiles):

```
LOAD * from 'lib://DataFiles/data1.csv';
```

라이브러리 연결에서 모든 구분된 데이터 파일 로드(DataFiles):

```
LOAD * from 'lib://DataFiles/*.csv';
```

쉼표를 구분 기호로 지정하고 포함된 레이블을 사용하여 구분된 파일 로드:

```
LOAD * from 'c:\userfiles\data1.csv' (ansi, txt, delimiter is ',', embedded labels);
```

탭을 구분 기호로 지정하고 포함된 레이블을 사용하여 구분된 파일 로드:

```
LOAD * from 'c:\userfiles\data2.txt' (ansi, txt, delimiter is '\t', embedded labels);
```

포함된 머리글과 함께 dif 파일 로드:

```
LOAD * from file2.dif (ansi, dif, embedded labels);
```

머리글 없이 고정 레코드 파일에서 3개의 필드 로드:

```
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);
```

절대 경로를 지정하여 QVX 파일 로드:

```
LOAD * from C:\qdssamples\xyz.qvx (qvx);
```

웹 파일 로드
웹 파일 데이터 연결에서 설정된 기본 URL에서 로드:

```
LOAD * from [lib://MyWebFile];
```

특정 URL에서 로드 및 웹 파일 데이터 연결에 설정된 URL 재정의:

```
LOAD * from [lib://MyWebFile] (URL is 'http://localhost:8000/foo.bar');
```

달러 기호 확장을 사용하여 변수에 설정된 특정 URL에서 로드:

```
SET dynamicURL = 'http://localhost/foo.bar';

LOAD * from [lib://MyWebFile] (URL is '$(dynamicURL)');
```

특정 필드 선택, 이름 변경 및 필드 계산
구분된 파일에서 3개의 특정 필드만 로드:

```
LOAD FirstName, LastName, Number from data1.csv;
```

레이블 없이 파일을 로드할 때 첫 번째 필드의 이름을 A로 변경하고 두 번째 필드의 이름을 B로 변경:

```
LOAD @1 as A, @2 as B from data3.txt (ansi, txt, delimiter is '\t', no labels);
```

FirstName, 공백 문자, LastName을 연결하여 Name 로드:

```
LOAD FirstName&' '&LastName as Name from data1.csv;
```

Quantity, Price 및 Value(제품의 Quantity 및 Price) 로드:

```
LOAD Quantity, Price, Quantity*Price as Value from data1.csv;
```

특정 레코드 선택
고유한 레코드만 로드하며 중복 레코드는 무시:

```
LOAD distinct FirstName, LastName, Number from data1.csv;
```

Litres 필드에 0을 넘는 값이 있는 레코드만 로드:

```
LOAD * from Consumption.csv where Litres>0;
```

파일에 없는 데이터 및 자동 생성된 데이터 로드
CatID 및 Category의 두 필드를 인라인 데이터로 테이블 로드:

```
LOAD * Inline

[CatID, Category

0,Regular

1,Occasional

2,Permanent];
```

UserID, Password 및 Access의 세 필드를 인라인 데이터로 테이블 로드:

```
LOAD * Inline [UserID, Password, Access

A, ABC456, User

B, VIP789, Admin];
```

10 000행이 있는 테이블 로드. A 필드에는 읽은 레코드 수(1,2,3,4,5...)가 포함되고 B 필드에는 0과 1 사이의 임의의 숫자가 포함됩니다.

```
LOAD RecNo( ) as A, rand( ) as B autogenerate(10000);
```

autogenerate 뒤에 오는 괄호는 허용되지만 필수는 아닙니다.

이전에 로드한 테이블에서 데이터 로드
먼저 구분된 테이블 파일을 로드한 후 tab1로 이름을 지정합니다.

```
tab1:

SELECT A,B,C,D from 'lib://DataFiles/data1.csv';
```

이미 로드된 tab1 테이블에서 필드를 tab2로 로드:

```
tab2:

LOAD A,B,month(C),A*B+D as E resident tab1;
```

이미 로드된 tab1 테이블에서 필드를 로드하지만 A가 B보다 큰 레코드만 로드:

```
tab3:

LOAD A,A+B+C resident tab1 where A>B;
```

이미 로드된 tab1 테이블에서 A로 정렬된 필드를 로드:

```
LOAD A,B*C as E resident tab1 order by A;
```

이미 로드된 tab1 테이블에서 필드를 첫 번째 필드, 두 번째 필드 순으로 정렬하여 로드:

```
LOAD A,B*C as E resident tab1 order by 1,2;
```

이미 로드된 tab1 테이블에서 필드를 C를 기준으로 내림차순 정렬한 후 B를 기준으로 오름차순 정렬한 다음 첫 번째 필드를 내림차순으로 정렬하여 로드:

```
LOAD A,B*C as E resident tab1 order by C desc, B asc, 1 desc;
```

이전에 로드한 필드에서 데이터 로드
이전에 로드한 Characters 테이블에서 Types 필드를 A로 로드:

```
LOAD A from_field (Characters, Types);
```

후속 테이블에서 데이터 로드(선행 LOAD)
A, B 및 계산 필드 X와 Y를 후속 **SELECT** 문으로 로드한 Table1에서 로드:

```
LOAD A, B, if(C>0,'positive','negative') as X, weekday(D) as Y;

SELECT A,B,C,D from Table1;
```

데이터 그룹화
ArtNo로 그룹화(집계)된 필드를 로드:

```
LOAD ArtNo, round(Sum(TransAmount),0.05) as ArtNoTotal from table.csv group by ArtNo;
```

Week 및 ArtNo로 그룹화(집계)된 필드를 로드:

```
LOAD Week, ArtNo, round(Avg(TransAmount),0.05) as WeekArtNoAverages from table.csv group by
Week, ArtNo;
```

하나의 레코드를 반복적으로 읽기
이 예에서는 한 필드에 요약된 각 학생의 성적이 포함된 입력 파일 Grades.csv를 사용합니다.

```
Student,Grades

Mike,5234

John,3345

Pete,1234

Paul,3352
```

1~5 눈금의 성적은 과목 Math, English, Science 및 History를 나타냅니다. **IterNo( )** 함수를 카운터로 사용하여 **while** 절로 여러 번 레코드를 읽으면 개별 값으로 성적을 구분할 수 있습니다. 읽은 각 성적은 **Mid** 함수로 추출되어 Grade에 저장됩니다. 과목은 **pick** 함수를 사용하여 선택되어 Subject에 저장됩니다. 마지막 **while** 절에는 다음 학생 레코드를 읽어야 함을 의미하는, 모든 성적을 읽었는지(이 경우는 학생당 4개) 확인하는 테스트 기능이 포함되어 있습니다.

```
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));
```

결과는 이 데이터가 포함된 테이블입니다.

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

분석 연결에서 로드
다음 샘플 데이터가 사용됩니다.

```
Values:
Load
 Rand() as A,
 Rand() as B,
 Rand() as C
AutoGenerate(50);
```

함수를 사용하여 데이터 로드

이 예제에서는 *Calculate(Parameter1, Parameter2)*라는 사용자 지정 함수가 포함된 *P*라는 이름의 분석 연결 플러그인이 있다고 가정합니다. 함수가 필드 *Field1*과 *Field2*가 포함된 테이블 *결과*를 반환합니다.

```
Load * Extension P.Calculate( Values{A, C} );
```
필드 A 및 C를 함수에 보낼 때 반환되는 모든 필드를 로드합니다.

```
Load Field1 Extension P.Calculate( Values{A, C} );
```
필드 A 및 C를 함수에 보낼 때 Field1 필드만 로드합니다.

```
Load * Extension P.Calculate( Values );
```
필드 A 및 B를 함수에 보낼 때 반환되는 모든 필드를 로드합니다. 필드가 지정되지 않았기 테이블에 나와 있는 순서에서 첫 번째인 A 및 B가 사용됩니다.

```
Load * Extension P.Calculate( Values {C, C});
```
필드 C를 함수의 두 매개 변수에 보낼 때 반환되는 모든 필드를 로드합니다.

```
Load * Extension P.Calculate( Values {String(A), Mixed(B)});
```
문자열로 강제 설정한 필드 A와 숫자로 강제 설정한 B를 함수에 보낼 때 반환되는 모든 필드를 로드합니다.

스크립트를 평가하여 데이터 로드

```
Load A as A_echo, B as B_echo Extension R.ScriptEval( 'q;', Values{A, B} );
```
A 및 B 값을 보낼 때 스크립트 q에서 반환되는 테이블을 로드합니다.

```
Load * Extension R.ScriptEval( '$(My_R_Script)', Values{A, B} );
```
A 및 B 값을 보낼 때 My_R_Script 변수에 저장된 스크립트에서 반환하는 테이블을 로드합니다.

```
Load * Extension R.ScriptEval( '$(My_R_Script)', Values{B as D, *} );
```
D, A 및 C로 이름을 바꾼 B 값을 보낼 때 My_R_Script 변수에 저장된 스크립트에서 반환하는 테이블을 로드합니다. *를 사용하면 참조되지 않은 나머지 필드를 보냅니다.

DataFiles 연결의 파일 확장명은 대/소문자를 구분합니다. 예를 들어 .qvd 입니다.

서식 사양 항목

각 서식 사양 항목은 테이블 파일의 특정 속성을 정의합니다.

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`]

문자 집합

문자 집합은 파일에 사용되는 문자 집합을 정의하는 **LOAD** 문의 파일 지정자입니다.

ansi, **oem** 및 **mac** 지정자가 QlikView에서 사용되었으며 여전히 작동합니다. 그러나 Qlik Sense를 사용하여 **LOAD** 문을 만든 경우에는 생성되지 않습니다.

구문:

```
utf8 | unicode | ansi | oem | mac | codepage is
```

인수:

인수

인수	설명
utf8	UTF-8 문자 집합
unicode	Unicode 문자 집합
ansi	Windows, 코드 페이지 1252
oem	DOS, OS/2, AS400 등
mac	코드 페이지 10000
codepage is	**codepage** 지정자를 사용하면 모든 Windows 코드 페이지를 *N* 으로 사용할 수 있습니다.

제한 사항:

macOS의 경우 **oem** 문자 집합에서 변환하는 기능이 구현되어 있지 않습니다. 아무것도 지정하지 않으면 Windows의 경우 코드 페이지 1252가 사용됩니다.

```
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)
```

관련 항목:

Load (page 152)

테이블 형식

테이블 형식은 파일 유형을 정의하는 **LOAD** 문을 위한 파일 지정자입니다. 아무것도 지정하지 않으면 *.txt* 파일이 지정된 것으로 가정합니다.

테이블 형식 유형

유형	설명
txt	구분 텍스트 파일에서 테이블의 열은 구분 기호에 의해 구분됩니다.

유형	설명
fix	고정 레코드 파일에서 각 필드는 정확히 일정한 수의 문자로 정해져 있습니다. 일반적으로 많은 고정 레코드 길이 파일은 줄 바꿈 문자로 구분된 레코드를 포함하고 있지만, 레코드 크기를 바이트 단위로 지정하거나 **Record is**를 사용하여 두 줄 이상으로 확장할 수 있는 고급 옵션도 있습니다. *데이터에 멀티바이트 문자가 포함된 경우 서식이 바이트 단위의 고정 길이를 기반으로 하기 때문에 필드 구분선이 정렬되지 않을 수 있습니다.*
dif	*.dif* 파일(Data Interchange Format)은 사용되는 테이블을 정의하는 데 사용되는 특수 형식입니다.
biff	또한 Qlik Sense는 *biff* 서식(Binary Interchange File Format)을 사용하여 표준 Excel 파일로 데이터를 해석할 수도 있습니다.
ooxml	Excel 2007 이상 버전은 ooxml *.xslx* 형식을 사용합니다.
html	테이블이 html 페이지 또는 파일의 일부인 경우 html을 사용해야 합니다.
xml	xml(Extensible Markup Language)은 데이터 구조를 텍스트 서식으로 표현하는 데 사용되는 공통 마크업 언어입니다.
qvd	*qvd* 형식은 Qlik Sense 앱에서 내보낸 독점 QVD 파일 형식입니다.
qvx	*qvx*는 Qlik Sense로의 고성능 출력을 수행하기 위한 파일/스트림 형식입니다.
parquet	Apache Parquet는 용량 데이터 집합을 저장하고 쿼리하는 데 매우 효율적인 열 기반 저장소 형식입니다.

Delimiter is

구분된 테이블 파일의 경우 **delimiter is** 지정자를 통해 임의의 구분 기호를 지정할 수 있습니다. 이 지정자는 구분된 .txt 파일에만 해당합니다.

구문:

```
delimiter is char
```

인수:

인수

인수	설명
char	127 ASCII 문자에서 1문자를 지정합니다.

또한 다음 값을 사용할 수도 있습니다.

선택적 값

값	설명
'\t'	인용 부호가 있거나 없는 탭 기호를 나타냅니다.
'\\'	백슬래시(\ ) 문자를 나타냅니다.
'spaces'	하나 이상의 공백의 모든 조합을 나타냅니다. CR 및 LF를 제외한 ASCII 값 32 미만의 인쇄 불가능한 문자는 공백으로 해석됩니다.

아무것도 지정하지 않으면 **delimiter is ','**이 사용됩니다.

```
LOAD * from a.txt (utf8, txt, delimiter is ',' , embedded labels);
```

관련 항목:

Load (page 152)

No eof

no eof 지정자는 구분된 **.txt** 파일을 로드할 때 파일 끝(EOF) 문자를 무시하기 위해 사용됩니다.

구문:

```
no eof
```

no eof 지정자를 사용하면 코드 포인트 26을 가진 문자(파일 끝(EOF))가 무시되고 필드 값의 일부가 될 수 있습니다.

이 지정자는 구분된 텍스트 파일에만 해당합니다.

```
LOAD * from a.txt (txt, utf8, embedded labels, delimiter is ' ', no eof);
```

관련 항목:

Load (page 152)

Labels

Labels는 **LOAD** 문에 대한 파일 지정자이며 파일에서 필드 이름의 위치를 정의합니다.

구문:

```
embedded labels|explicit labels|no labels
```

필드 이름은 파일의 다양한 위치에 있을 수 있습니다. 첫 번째 레코드에 필드 이름이 포함되는 경우 **embedded labels**를 사용해야 합니다. 필드 이름을 찾을 수 없는 경우 **no labels**를 사용해야 합니다. *dif* 파일에는 명시적 필드 이름이 포함된 별도의 머리글 섹션이 사용되는 경우도 있습니다. 이 경우 **explicit labels**를 사용해야 합니다. *dif* 파일의 경우에도 아무것도 지정하지 않으면 **embedded labels**가 사용됩니다.

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)
```

관련 항목:

Load (page 152)

Header is

테이블 파일의 머리글 크기를 지정합니다. **header is** 지정자로 임의의 머리글 길이를 지정할 수 있습니다. 머리글은 Qlik Sense에 사용되지 않는 텍스트 섹션입니다.

구문:

```
header is n
```

```
header is line
```

```
header is n lines
```

머리글 길이는 바이트(**header is n**) 또는 줄(**header is line** 또는 **header is n lines**) 단위로 지정할 수 있습니다. **n**은 머리글 길이를 나타내는 양의 정수여야 합니다. 지정하지 않으면 **header is 0**이 지정된 것으로 가정합니다. **header is** 지정자는 테이블 파일에만 사용할 수 있습니다.

다음은 Qlik Sense에서 데이터로 해석하지 않아야 하는 머리글 텍스트 행이 포함된 데이터 소스 테이블의 예입니다.

```
*Header line
Col1,Col2
a,B
c,D
```

header is 1 lines 지정자를 사용하면 첫 번째 행은 데이터로 로드되지 않습니다. 이 예에서는 **embedded labels** 지정자를 사용하여 Qlik Sense에서 제외되지 않은 첫 번째 행에 필드 레이블이 포함된 것으로 해석하도록 지정합니다.

```
LOAD Col1, Col2
FROM 'lib://files/header.txt'
(txt, embedded labels, delimiter is ',', msq, header is 1 lines);
```

결과로는 Col1 및 Col2의 두 필드가 포함된 테이블이 생성됩니다.

관련 항목:

- *Load (page 152)*

Record is

고정 레코드 길이 파일의 경우 **record is** 지정자를 통해 레코드 길이를 지정해야 합니다.

구문:

```
Record is n
Record is line
Record is n lines
```

인수:

인수

인수	설명
n	레코드 길이를 바이트 단위로 지정합니다.
line	레코드 길이를 1행으로 지정합니다.
n lines	행 수로 레코드 길이를 지정하며, 여기서 n은 레코드 길이를 나타내는 양의 정수입니다.

제한 사항:

record is 지정자는 **fix** 파일에만 사용할 수 있습니다.

관련 항목:

- *Load (page 152)*

Quotes

Quotes는 **LOAD** 문에 대한 파일 지정자로, 인용 부호의 사용 가능 여부 및 인용 부호와 구분 기호 간의 우선 순위를 정의합니다. 텍스트 파일에서만 사용할 수 있습니다.

구문:

```
no quotes
```

```
msq
```

지정자가 생략된 경우에는 표준 인용 부호, 즉 " " 또는 ' '같은 인용 부호가 사용됩니다. 단, 이 인용 부호가 필드 값의 첫 번째와 마지막에 오는 공백이 아닌 문자여야 합니다.

인수:

인수

인수	설명
no quotes	텍스트 파일에서 인용 부호를 사용할 수 없는 경우에 사용됩니다.
msq	필드에 여러 줄로 된 내용을 허용하는 최신 스타일 인용 부호를 지정하는 데 사용됩니다. 줄의 끝 문자를 포함하는 필드는 반드시 큰따옴표로 묶어야 합니다. msq 옵션 사용 시 한 가지 제한 사항은 필드 내용에 첫 번째 또는 마지막 문자로 한 개의 큰따옴표(") 문자가 표시되면 여러 줄로 된 내용의 시작 또는 끝으로 해석되어 로드된 데이터 셋에 예상하지 못한 결과를 초래할 수 있다는 점입니다. 이 경우에는 지정자를 생략하고 표준 인용 부호를 사용해야 합니다.

XML

이 스크립트 지정자는 xml 파일을 로드할 때 사용됩니다. **XML** 지정자의 유효한 옵션이 구문에 나열됩니다.

Qlik Sense에서 DTD 파일을 로드할 수 없습니다.

구문:

```
xmlsimple
```

관련 항목:

Load (page 152)

KML

이 스크립트 지정자는 KML 파일을 로드하여 맵 시각화에 사용하는 경우에 사용됩니다.

구문:

```
kml
```

KML 파일은 다각형으로 표시되는 영역 데이터(예: 국가 및 지역), 선 데이터(예: 추적 또는 로드) 또는 [경도, 위도] 형식의 점으로 표시되는 포인트 데이터(예: 도시 또는 장소)를 표시할 수 있습니다.

URL is

이 스크립트 지정자는 웹 파일을 로드할 때 웹 파일 데이터 연결의 URL을 설정하는 데 사용됩니다.

구문:

```
URL is string
```

인수:

인수

인수	설명
string	로드할 파일의 URL을 지정합니다. 이렇게 하면 사용되는 웹 파일 연결에 설정된 URL이 재정의됩니다.

제한 사항:

URL is 지정자는 웹 파일과만 관련됩니다. 기존 웹 파일 데이터 연결을 사용해야 합니다.

관련 항목:

Load (page 152)

userAgent is

이 스크립트 지정자는 웹 파일을 로드할 때 브라우저 사용자 에이전트를 설정하는 데 사용됩니다.

구문:

```
userAgent is string
```

인수:

인수

인수	설명
string	브라우저 사용자 에이전트 문자열을 지정합니다. 이렇게 하면 기본 브라우저 사용자 에이전트 "Mozilla/5.0"이 재정의됩니다.

제한 사항:

userAgent is 지정자는 웹 파일과만 관련됩니다.

관련 항목:

Load (page 152)

Let

let 문은 스크립트 변수 정의에 사용된다는 점에서 **set** 문과 차이가 있습니다. **let** 문은 **set** 문과는 반대로 변수에 할당하기 전 스크립트 런타임에 '='의 오른쪽에 있는 표현식을 평가합니다.

구문:

```
Let variablename=expression
```

예 및 결과:

예	결과

`Set x=3+4;`
 	
`Let y=3+4;`
 	
`z=$(y)+1;`
	`$(x)`는 '3+4'로 평가됩니다.
	
	`$(y)`는 ' 7'로 평가됩니다.
	
	`$(z)`는 ' 8'로 평가됩니다.
	
	Set 및 **Let** 문 사이의 차이점에 유의하십시오. **Set** 문은 문자열 '3+4'를 변수에 할당하는 반면 **Let** 문은 문자열을 평가하고 변수에 7을 할당합니다.

Let T=now( );	`$(T)`에는 현재 시간 값이 지정됩니다.

Loosen Table

Loosen Table 문을 사용하면 스크립트 실행 중 하나 이상의 Qlik Sense 내부 데이터 테이블이 명시적으로 느슨하게 결합되도록 선언될 수 있습니다. 테이블이 느슨하게 결합된 경우 테이블 내 필드 값 간의 모든 연결이 제거됩니다. 느슨하게 결합된 테이블의 각 필드를 연결되지 않은 별도의 테이블로 로드하면 비슷한 효과를 낼 수 있습니다. 느슨한 결합은 테스트 중에 데이터 구조의 다른 부분을 임시로 격리할 때 유용할 수 있습니다. 느슨하게 결합된 테이블은 테이블 뷰어에서 점선으로 표시됩니다. 스크립트에서 **Loosen Table** 문을 하나 이상 사용하면 Qlik Sense가 스크립트 실행 전에 생성된 느슨하게 결합된 테이블 설정을 모두 무시하게 됩니다.

구문:

```
Loosen Tabletablename [ , tablename2 ...]
```

```
Loosen Tablestablename [ , tablename2 ...]
```

Loosen Table 또는 **Loosen Tables** 구문을 사용할 수 있습니다.

> *Qlik Sense가 데이터 구조 내에서 스크립트에 대화식으로 또는 명시적으로 느슨하게 결합되도록 선언된 테이블로 분리할 수 없는 순환 참조를 찾은 경우 순환 참조가 모두 없어질 때까지 하나 이상의 추가 테이블이 강제로 느슨하게 결합됩니다. 그러면 **루프 경고** 대화 상자가 표시됩니다.*

```
Tab1:

SELECT * from Trans;

Loosen Table Tab1;
```

Map

map ... using 문은 특정 필드 값 또는 표현식을 특정 매핑 테이블의 값에 매핑하는 데 사용됩니다. 매핑 테이블은 **Mapping** 문을 통해 만듭니다.

구문:

```
Map fieldlist Using  mapname
```

자동 매핑은 스크립트의 끝 또는 **Unmap** 문에 도달할 때까지 **Map ... Using** 문 이후 로드된 필드에 대해 수행됩니다.

매핑은 Qlik Sense의 내부 테이블에 필드가 저장되기 전 일련의 이벤트에서 마지막으로 수행됩니다. 즉, 매핑은 표현식 실행 도중 필드 이름이 발견될 때마다 수행되는 것이 아니라 내부 테이블의 필드 이름으로 값이 저장될 때 수행됩니다. 표현식 수준에서 매핑이 필요한 경우 **Applymap()** 함수를 대신 사용해야 합니다.

인수:

인수

인수	설명
fieldlist	스크립트의 이 지점에서 매핑할 필드의 쉼표로 구분된 목록입니다. 필드 목록에 *를 사용하면 모든 필드를 지정할 수 있습니다. 필드 이름에는 와일드카드 문자 * 및 ?가 허용됩니다. 와일드카드를 사용할 경우 필드 이름을 따옴표로 묶어야 합니다.
mapname	이전에 **mapping load** 또는 **mapping select** 문으로 읽은 매핑 테이블의 이름입니다.

예 및 결과:

예	결과
Map Country Using Cmap;	map Cmap을 사용하여 Country 필드를 매핑할 수 있습니다.
Map A, B, C Using X;	map X를 사용하여 A, B 및 C 필드를 매핑할 수 있습니다.
Map * Using GenMap;	GenMap을 사용하여 모든 필드를 매핑할 수 있습니다.

NullAsNull

NullAsNull 문은 이전에 **NullAsValue** 문으로 설정된 문자열 값으로의 NULL 값 변환을 해제합니다.

구문:

```
NullAsNull *fieldlist
```

NullAsValue 문은 스위치로 작동하며 스크립트에서 **NullAsValue** 또는 **NullAsNull** 문을 사용하여 여러 번 설정하거나 해제할 수 있습니다.

인수:

인수

인수	설명
*fieldlist	**NullAsNull**을 설정할 필드의 쉼표로 구분된 목록입니다. 필드 목록에 *를 사용하면 모든 필드를 지정할 수 있습니다. 필드 이름에는 와일드카드 문자 * 및 ?가 허용됩니다. 와일드카드를 사용할 경우 필드 이름을 따옴표로 묶어야 합니다.

```
NullAsNull A,B;
LOAD A,B from x.csv;
```

NullAsValue

NullAsValue 문은 NULL을 값으로 변환해야 하는 필드를 지정합니다.

구문:

```
NullAsValue *fieldlist
```

기본적으로 Qlik Sense에서는 NULL 값을 누락 항목 또는 정의되지 않은 항목으로 간주합니다. 하지만 특정 데이터베이스 컨텍스트에서는 NULL 값을 단순히 누락된 값이 아닌 특별한 값으로 간주할 것임을 나타냅니다. **NullAsValue** 문을 사용하면 일반적으로 NULL 값은 다른 NULL 값과 연결할 수 없다는 사실 적용이 일시 중지됩니다.

NullAsValue 문은 스위치 역할을 하며 후속 로드 문에서 작동합니다. **NullAsNull** 문을 사용하면 이 문을 다시 해제할 수 있습니다.

인수:

인수

인수	설명
*fieldlist	**NullAsValue**을 설정할 필드의 쉼표로 구분된 목록입니다. 필드 목록에 *를 사용하면 모든 필드를 지정할 수 있습니다. 필드 이름에는 와일드카드 문자 * 및 ?가 허용됩니다. 와일드카드를 사용할 경우 필드 이름을 따옴표로 묶어야 합니다.

```
NullAsValue A,B;
Set NullValue = 'NULL';
LOAD A,B from x.csv;
```

Qualify

Qualify 문은 필드 이름의 한정을 설정하는 데 사용됩니다. 즉, 필드 이름 앞에 테이블 이름이 추가됩니다.

구문:

```
Qualify *fieldlist
```

테이블 이름으로 필드 이름을 한정하는 **qualify** 문에 의해 다른 테이블에서 동일한 이름을 갖는 필드 간의 자동 조인이 일시 중지될 수 있습니다. 한정된 필드가 테이블에 있는 경우 필드 이름이 변경됩니다. 새 이름은 *tablename.fieldname*의 형식으로 작성됩니다. *Tablename*은 현재 테이블의 레이블과 동일하거나, 레이블이 없는 경우는 **LOAD** 및 **SELECT** 문에서 **from** 다음에 표시되는 이름과 동일합니다.

qualify 문 다음에 로드되는 모든 필드에 대해 한정이 적용됩니다.

한정은 기본적으로 스크립트 실행 시작 시 항상 해제되어 있습니다. **qualify** 문을 사용하면 언제든 필드 이름의 한정을 활성화할 수 있습니다. **Unqualify** 문을 사용하면 언제든 한정을 해제할 수 있습니다.

***qualify** 문은 부분 재로드와 함께 사용하지 않도록 해야 합니다.*

인수:

인수

인수	설명
*fieldlist	한정을 설정할 필드의 쉼표로 구분된 목록입니다. 필드 목록에 *를 사용하면 모든 필드를 지정할 수 있습니다. 필드 이름에는 와일드카드 문자 * 및 ?가 허용됩니다. 와일드카드를 사용할 경우 필드 이름을 따옴표로 묶어야 합니다.

Example 1:

```
Qualify B;

LOAD A,B from x.csv;

LOAD A,B from y.csv;
```

두 테이블 **x.csv** 및 **y.csv**는 **A**를 통해서만 연결됩니다. 결과는 다음 세 필드입니다. A, x.B, y.B.

Example 2:

친숙하지 않은 데이터베이스에서는 다음 예에 설명된 것과 같이 하나의 필드만 또는 일부 필드만 연결하고 시작하는 것이 유용할 수 있습니다.

```
qualify *;

unqualify TransID;

SQL SELECT * from tab1;

SQL SELECT * from tab2;

SQL SELECT * from tab3;
```

tab1, *tab2* 및 *tab3* 테이블 간의 연결에는 **TransID**만 사용됩니다.

Rem

rem 문은 스크립트에 설명 또는 주석을 삽입하거나 스크립트 문을 제거하지 않고 일시적으로 비활성화할 때 사용됩니다.

구문:

```
Rem string
```

rem 문과 다음에 오는 세미콜론(**;**) 사이의 모든 문자는 주석으로 간주합니다.

다음과 같은 두 가지 대체 방법을 사용하여 스크립트에 주석을 달 수도 있습니다.

1. 두 인용 부호 사이를 제외하고 스크립트의 어느 곳에라도 **/*** 및 ***/** 사이에 해당 섹션을 넣는 방법으로 주석을 달 수 있습니다.
2. 스크립트에 **//** 기호를 입력하면 동일한 행에서 기호의 오른쪽에 위치하는 모든 텍스트가 주석으로 처리됩니다. (//:은 예외적으로 인터넷 주소의 일부로 사용될 수도 있습니다.)

인수:

인수

인수	설명
string	임의의 텍스트입니다.

```
Rem ** This is a comment **;
/* This is also a comment */
// This is a comment as well
```

Rename

Rename 스크립트 키워드는 이미 로드된 테이블 또는 필드의 이름을 변경하는 데 사용할 수 있습니다.

Rename field

이 스크립트 함수는 기존 Qlik Sense 필드를 하나 이상 로드한 후 해당 필드의 이름을 바꿉니다.

Qlik Sense의 필드 또는 함수와 동일하게 변수 이름을 지정하지 않는 것이 좋습니다.

rename field 또는 **rename fields** 구문을 사용할 수 있습니다.

구문:

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

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

인수:

인수	설명
mapname	이전 필드 이름과 새 필드 이름 쌍을 하나 이상 포함하고 있는, 이전에 로드한 매핑 테이블의 이름입니다.
oldname	이전 필드 이름입니다.
newname	새 필드 이름입니다.

제한 사항:

두 필드가 동일한 이름을 갖도록 이름을 바꿀 수는 없습니다.

Example 1:

```
Rename Field XAZ0007 to Sales;
```

Example 2:

```
FieldMap:

Mapping SQL SELECT oldnames, newnames from datadictionary;

Rename Fields using FieldMap;
```

Rename table

이 스크립트 함수는 기존 Qlik Sense 내부 테이블을 하나 이상 로드한 후 해당 테이블의 이름을 바꿉니다.

rename table 또는 **rename tables** 구문을 사용할 수 있습니다.

구문:

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

인수:

인수

인수	설명
mapname	이전 테이블 이름과 새 테이블 이름 쌍을 하나 이상 포함하고 있는, 이전에 로드한 매핑 테이블의 이름입니다.
oldname	이전 테이블 이름입니다.
newname	새 테이블 이름입니다.

제한 사항:

서로 다른 이름을 가진 두 테이블의 이름을 동일한 이름으로 바꿀 수는 없습니다. 테이블의 이름을 기존 테이블과 동일한 이름으로 바꾸려고 하면 스크립트에서 오류가 발생합니다.

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

Search 문은 스마트 검색에서 필드를 포함 또는 제외하는 데 사용됩니다.

구문:

```
Search Include *fieldlist
Search Exclude *fieldlist
```

여러 Search 문을 사용하여 포함시킬 필드를 구체적으로 선택할 수 있습니다. 이 문은 맨 위에서 아래로 평가됩니다.

인수:

인수

인수	설명
*fieldlist	스마트 검색 시 검색에 추가 또는 제외시킬 쉼표로 구분된 필드 목록입니다. 필드 목록에 * 를 사용하면 모든 필드를 지정할 수 있습니다. 필드 이름에는 와일드카드 문자 * 및 ?가 허용됩니다. 와일드카드를 사용할 경우 필드 이름을 따옴표로 묶어야 합니다.

검색 예

문	설명
`Search Include *;`	스마트 검색 시 모든 필드를 검색에 포함시킵니다.
`Search Exclude [*ID];`	스마트 검색 시 ID로 끝나는 모든 필드를 검색에서 제외시킵니다.
`Search Exclude '*ID';`	스마트 검색 시 ID로 끝나는 모든 필드를 검색에서 제외시킵니다.
`Search Include ProductID;`	스마트 검색 시 ProductID 필드를 검색에 포함시킵니다.

이 시퀀스에서 이들 3개 문이 결합된 결과는 ID(ProductID 제외)로 끝나는 모든 필드가 스마트 검색 시 검색에서 제외되는 것입니다.

Section

section 문을 사용하면 후속 **LOAD** 및 **SELECT** 문을 데이터 또는 액세스 권한의 정의로 고려할지 정의할 수 있습니다.

구문:

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

아무것도 지정하지 않으면 **section application**이 사용됩니다. **section** 정의는 새 **section** 문을 만들 때까지 유효합니다.

```
Section access;

Section application;
```

Select

ODBC 데이터 소스 또는 OLE DB 제공자의 필드는 표준 SQL **SELECT** 문을 통해 선택됩니다. 그러나 **SELECT** 문의 허용 여부는 사용되는 ODBC 드라이버 또는 OLE DB 제공자에 의해 좌우됩니다. **SELECT** 문을 사용하려면 소스에 대한 개방형 데이터 연결이 필요합니다.

구문:

```
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 ]
```

또한 여러 개의 **SELECT** 문은 **union** 연산자를 사용하여 하나로 연결할 수도 있습니다.

```
selectstatement Union selectstatement
```

SELECT 문은 ODBC 드라이버 또는 OLE DB 공급자에 의해 해석되므로 ODBC 드라이버 또는 OLE DB 공급자의 기능에 따라 일반 SQL 구문과의 편차가 발생할 수 있습니다. 예를 들어 보면 다음과 같습니다.

- **as**는 허용되지 않는 경우가 많습니다. 즉, *aliasname*이 *fieldname* 바로 다음에 나와야 합니다.
- *aliasname*이 사용된 경우 **as**가 필수일 수 있습니다.
- **distinct**, **as**,**where**, **group by**, **order by** 또는 **union**은 지원되지 않는 경우가 있습니다.
- ODBC 드라이버는 위에 나와 있는 다양한 인용 부호를 모두 허용하지 않을 수 있습니다.

*위의 내용은 SQL **SELECT** 문에 대한 완전한 설명이 아닙니다. 예를 들어 **SELECT** 문은 중첩될 수 있고 한 **SELECT** 문에 여러 조인을 만들 수 있으며, 표현식에서 허용되는 함수가 매우 많을 수도 있습니다.*

인수:

인수

인수	설명
distinct	**distinct**는 선택된 필드의 중복 값 조합이 한 번만 로드되어야 하는 경우 사용되는 조건자입니다.
distinctrow	**distinctrow**는 소스 테이블의 중복 레코드가 한 번만 로드되어야 하는 경우 사용되는 조건자입니다.
fieldlist	**fieldlist ::= (*| field) {, field }** 선택할 필드의 목록입니다. 필드 목록에 *를 사용하면 테이블의 모든 필드를 지정할 수 있습니다. **fieldlist ::= field {, field }** 쉼표로 구분된 하나 이상의 필드 목록입니다. **field ::= (fieldref| expression) [as aliasname]** expression은 하나 또는 여러 다른 필드에 기반을 둔 숫자 또는 문자열 함수일 수 있습니다. 일반적으로 허용되는 일부 연산자 및 함수에는 +, -, *, /, &(문자열 연결), sum(fieldname), count(fieldname), avg(fieldname)(average), month(fieldname) 등이 있습니다. 자세한 내용은 ODBC 드라이버의 설명서를 참조하십시오. **fieldref ::= [tablename.] fieldname** **tablename** 및 **fieldname**은 암시하는 것과 동일한 텍스트 문자열입니다. 공백이 포함된 경우 곧은 큰따옴표로 묶어야 합니다. **as** 절은 필드에 새 이름을 할당하는 데 사용합니다.
from	**tablelist ::= table {, table }** 필드를 선택할 테이블의 목록입니다. **table ::= tablename [[as] aliasname]** **tablename**은 따옴표로 묶어도 되고 묶지 않아도 됩니다.
where	**where**는 레코드를 선택에 포함할지를 나타내는 데 사용하는 절입니다. **criterion**은 때때로 매우 복잡할 수 있는 논리 표현식입니다. 허용되는 일부 연산자에는 숫자 연산자와 함수, **=**, **<>** 또는 **#**(같지 않음), **>**, **>=**, **<**, **<=**, **and**, **or**, **not**, **exists**, **some**, **all**, **in** 및 새로운 **SELECT** 문이 있습니다. 자세한 내용은 ODBC 드라이버 또는 OLE DB 공급자의 문서를 참조하십시오.

인수	설명
group by	**group by**는 여러 레코드를 하나로 집계(그룹화)하는 데 사용되는 절입니다. 특정 필드에서 한 그룹에 속하는 모든 레코드는 동일한 값을 가져야 하며 그렇지 않을 경우 해당 필드는 한 표현식(예: sum 또는 average) 내에서만 사용될 수 있습니다. 하나 또는 여러 필드에 기반을 둔 표현식은 필드 기호의 표현식으로 정의됩니다.
having	**having**은 **where** 절이 레코드 정규화에 사용되는 방식과 유사하게 그룹을 정규화하는 데 사용되는 절입니다.
order by	**order by**는 **SELECT** 문의 결과 테이블의 정렬 순서를 지정하는 데 사용되는 절입니다.
join	**join**은 여러 개의 테이블을 하나로 합칠 것인지 지정하는 한정자입니다. 필드 이름 및 테이블 이름에 문자 집합의 공백 또는 문자가 포함된 경우 따옴표로 묶어야 합니다. 스크립트가 Qlik Sense에 의해 자동으로 생성된 경우 사용된 인용 부호는 **Connect** 문에서 데이터 소스의 데이터 소스 정의에 지정된 ODBC 드라이버 또는 OLE DB 공급자에 의해 기본 설정된 부호입니다.

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

set 문은 스크립트 변수를 정의하는 데 사용됩니다. 문자열, 경로, 드라이브 등을 대체하는 데 사용할 수 있습니다.

구문:

```
Set variablename=string
```

Example 1:

```
Set FileToUse=Data1.csv;
```

Example 2:

```
Set Constant="My string";
```

Example 3:

```
Set BudgetYear=2012;
```

Sleep

sleep 문은 지정된 시간 동안 스크립트 실행을 일시 중지합니다.

구문:

```
Sleep n
```

인수:

인수	설명
n	밀리초로 지정되며, 여기서 *n*은 3600000(즉, 1시간)보다 크지 않은 양의 정수입니다. 이 값은 표현식일 수 있습니다.

Example 1:

```
Sleep 10000;
```

Example 2:

```
Sleep t*1000;
```

SQL

SQL 문을 사용하면 ODBC 또는 OLE DB 연결을 통해 임의의 SQL 명령을 전송할 수 있습니다.

구문:

```
SQL sql_command
```

Qlik Sense에서 읽기 전용 모드로 ODBC 연결을 연 경우 데이터베이스를 업데이트하는 SQL 문을 전송하면 오류가 반환됩니다.

다음 구문을 보십시오.

```
SQL SELECT * from tab1;
```

이 구문은 사용 가능하며 일관성을 위해 **SELECT**의 기본 구문으로 사용됩니다. 하지만 SQL 접두사는 **SELECT** 문에 대한 옵션으로 유지됩니다.

인수:

인수	설명
sql_command	유효한 SQL 명령입니다.

Example 1:

```
SQL leave;
```

Example 2:

```
SQL Execute <storedProc>;
```

SQLColumns

sqlcolumns 문은 **connect**가 설정된 ODBC 또는 OLE DB 데이터 소스의 열을 나타내는 필드 집합을 반환합니다.

구문:

```
SQLcolumns
```

이 필드는 주어진 데이터베이스에 대한 유용한 개요를 제공하기 위해 **sqltables** 및 **sqltypes** 명령으로 생성된 필드와 결합될 수 있습니다. 12가지 표준 필드는 다음과 같습니다.

TABLE_QUALIFIER

TABLE_OWNER

TABLE_NAME

COLUMN_NAME

DATA_TYPE

TYPE_NAME

PRECISION

LENGTH

SCALE

RADIX

NULLABLE

REMARKS

이러한 필드에 대한 자세한 설명은 ODBC 참조 안내서를 참조하십시오.

```
Connect to 'MS Access 7.0 Database; DBQ=C:\Course3\DataSrc\QWT.mbd';
SQLcolumns;
```

일부 ODBC 드라이버는 이 명령을 지원하지 않을 수 있습니다. 일부 ODBC 드라이버는 추가 필드를 생성할 수 있습니다.

SQLTables

sqltables 문은 **connect**가 설정된 ODBC 또는 OLE DB 데이터 소스의 테이블을 나타내는 필드 집합을 반환합니다.

구문:

```
SQLTables
```

이 필드는 주어진 데이터베이스에 대한 유용한 개요를 제공하기 위해 **sqlcolumns** 및 **sqltypes** 명령으로 생성된 필드와 결합될 수 있습니다. 5가지 표준 필드는 다음과 같습니다.

TABLE_QUALIFIER

TABLE_OWNER

TABLE_NAME

TABLE_TYPE

REMARKS

이러한 필드에 대한 자세한 설명은 ODBC 참조 안내서를 참조하십시오.

```
Connect to 'MS Access 7.0 Database; DBQ=C:\Course3\DataSrc\QWT.mbd';
SQLTables;
```

일부 ODBC 드라이버는 이 명령을 지원하지 않을 수 있습니다. 일부 ODBC 드라이버는 추가 필드를 생성할 수 있습니다.

SQLTypes

sqltypes 문은 **connect**가 설정된 ODBC 또는 OLE DB 데이터 소스의 유형을 나타내는 필드 집합을 반환합니다.

구문:

```
SQLTypes
```

이 필드는 주어진 데이터베이스에 대한 유용한 개요를 제공하기 위해 **sqlcolumns** 및 **sqltables** 명령으로 생성된 필드와 결합될 수 있습니다. 15가지 표준 필드는 다음과 같습니다.

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

이러한 필드에 대한 자세한 설명은 ODBC 참조 안내서를 참조하십시오.

```
Connect to 'MS Access 7.0 Database; DBQ=C:\Course3\DataSrc\QWT.mbd';
SQLTypes;
```

일부 ODBC 드라이버는 이 명령을 지원하지 않을 수 있습니다. 일부 ODBC 드라이버는 추가 필드를 생성할 수 있습니다.

Star

데이터베이스의 모든 필드 값 집합을 표시하기 위해 사용되는 문자열은 **star** 문을 통해 설정할 수 있습니다. 후속 **LOAD** 문 및 **SELECT** 문에 영향을 줍니다.

구문:

```
Star is[ string ]
```

인수:

인수

인수	설명
string	임의의 텍스트입니다. 문자열에 공백이 포함된 경우 인용 부호로 묶어야 합니다. 아무것도 지정하지 않으면 **star is;**가 사용됩니다. 즉, 명시적으로 지정된 경우가 아니면 별표 기호를 사용할 수 없습니다. 이 정의는 새 **star** 문을 만들 때까지 유효합니다.

Star is 문은 Section Access를 사용하는 경우 스크립트의 데이터 부분(**Section Application** 아래)에 사용하지 않는 것이 좋습니다. 그러나 스크립트의 **Section Access** 부분의 보호되는 필드에서는 별표 문자가 완전히 지원됩니다. 이 경우 명시적 **Star is** 문을 사용할 필요가 없습니다. 섹션 액세스에서 항상 암시적이기 때문입니다.

제한 사항

- 테이블을 연결하는 필드인 키 필드에는 별표 문자를 사용할 수 없습니다.
- **Unqualify** 문의 영향을 받는 모든 필드에는 별표 문자를 사용할 수 없습니다. 테이블을 연결하는 필드에 영향을 줄 수 있기 때문입니다.
- 비논리 테이블(예: info-load 테이블 또는 mapping-load 테이블)에서는 별표 문자를 사용할 수 없습니다.
- 별표 문자가 섹션 액세스에서 축소 필드(데이터를 연결하는 필드)에 사용된 경우 섹션 액세스에서 이 필드에 나열된 값을 나타냅니다. 데이터에 존재할 수 있지만 섹션 액세스에 나열되지 않은 다른 값을 나타내지 않습니다.
- **Section Access** 영역 외의 모든 형태의 데이터 축소에 영향을 받는 필드에는 별표 문자를 사용할 수 없습니다.

예

아래 예는 Section Access를 수행하는 데이터 로드 스크립트에서 추출한 것입니다.

```
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

];
```

다음이 적용됩니다.

- *Star* 기호는 *입니다.
- 사용자 *ADMIN* 은 모든 필드를 볼 수 있습니다. 아무 것도 생략되지 않습니다.
- 사용자 *USER1* 는 *SALES* 필드를 볼 수 없습니다.
- 사용자 *USER2* 는 *WAREHOUSE* 필드를 볼 수 없습니다.
- 사용자 *USER3* 은 *EMPLOYEES* 필드를 볼 수 없습니다.
- 사용자 *USER4* 는 이 사용자의 두 필드인 *SALES* 및 *WAREHOUSE*를 생략(OMIT)하는 솔루션에 두 번 추가되어 있습니다.
- *USER5* 는 OMIT에서 나열된 모든 필드를 의미하는 "*"를 추가했습니다. 즉, 사용자 *USER5* 는 *SALES*, *WAREHOUSE* 및 *EMPLOYEES* 필드를 볼 수 없지만 *ORDERS* 필드는 볼 수 있습니다.

Store

Store 문은 QVD, Parquet, CSV 또는 TXT 파일을 만듭니다.

구문:

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

이 문은 명시적으로 명명된 QVD, Parquet 또는 텍스트 파일을 만듭니다.

또한 이 문은 하나의 데이터 테이블에서만 필드를 내보낼 수 있습니다. 여러 테이블의 필드를 내보내는 경우 내보낼 데이터 테이블을 만들려면 스크립트에서 미리 명시적 join을 수행해야 합니다.

텍스트 값은 UTF-8 형식의 CSV 파일로 내보냅니다. 구분 기호를 지정할 수 있습니다. 자세한 내용은 **LOAD**를 참조하십시오. CSV 파일에 대한 **store** 문은 BIFF 내보내기를 지원하지 않습니다.

형식이 올바르지 않은 데이터의 경우 데이터가 올바르게 해석되도록 필드를 큰따옴표로 묶습니다. 예를 들어, 필드에 따옴표, 쉼표, 공백 또는 줄 바꿈과 같은 문자가 포함된 경우 이런 일이 발생할 수 있습니다.

인수:

Store 명령 인수

인수	설명
*fieldlist::= (* | field) { , field })*	선택할 필드의 목록입니다. 필드 목록에 *를 사용하면 모든 필드를 지정할 수 있습니다. *field::= fieldname [***as** *aliasname]* *fieldname*은 *table*의 필드 이름과 동일한 텍스트입니다. (필드 이름에 공백 또는 기타 비표준 문자가 포함된 경우 큰따옴표 또는 대괄호로 묶어야 합니다.) *aliasname*은 생성된 QVD 또는 CSV 파일에서 사용할 필드의 대체 이름입니다.
table	데이터 소스로 사용하기 위해 이미 로드된 테이블을 나타내는 스크립트 레이블입니다.
filename	기존 폴더 데이터 연결에 대한 유효한 경로가 포함된 대상 파일의 이름입니다. ***'lib://Table Files/target.qvd'*** 레거시 스크립팅 모드에서는 다음 경로 형식도 지원됩니다. • 절대 경로 ***c:\data\sales.qvd*** • Qlik Sense 앱 작업 디렉터리에 대한 상대 경로. ***data\sales.qvd*** 경로를 생략한 경우 Qlik Sense는 **Directory** 문으로 지정된 디렉터리에 파일을 저장합니다. **Directory** 문이 없으면 Qlik Sense는 작업 디렉터리인 *C:\Users\{user}\Documents\Qlik\Sense\Apps*에 파일을 저장합니다. •

인수	설명
format-spec ::=((**txt** | **qvd** | **parquet**), **compression is** *코덱*)	서식 사양을 이러한 파일 형식 중 하나로 설정할 수 있습니다. 서식 사양을 생략하면 **qvd**가 사용됩니다. • CSV 및 TXT 파일의 경우 **txt**입니다. • QVD 파일의 경우 **qvd**. • Parquet 파일의 경우 **parquet**. **parquet**를 사용하는 경우 **compression is**와 함께 사용할 압축 코덱도 설정할 수 있습니다. **compression is**로 압축 코덱을 지정하지 않으면 snappy가 사용됩니다. 다음 압축 설정을 사용할 수 있습니다. • uncompressed • snappy • gzip • lz4 • brotli • zstd • lz4_hadoop 예: `Store mytable into [lib://DataFiles/myfile.parquet] (parquet, compression is lz4);`

```
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';
```

DataFiles 연결의 파일 확장명은 대/소문자를 구분합니다. 예를 들어 .qvd입니다.

Parquet 파일에 저장

Parquet는 각 필드에 단일 특정 유형의 데이터(예: in32, double, 타임스탬프 또는 텍스트)가 포함된 강력한 유형의 파일 형식입니다. Qlik Sense는 내부 데이터를 느슨한 유형의 이중으로 저장합니다. 여기서 서로 다른 소스의 데이터가 동일한 필드에 혼합될 수 있습니다. Parquet의 각 필드에는 이중의 한 부분만 저장할 수 있으므로 각 필드에 무엇이 포함되어 있는지 알아야 합니다. 기본적으로 Qlik Sense는 필드 유형을 사용하여 필드 저장 방법을 결정합니다. Parquet 파일에 데이터를 특정 형식으로 저장하는 경우 필드를 로드할 때 필드의 데이터 유형을 지정해야 합니다. 텍스트 필드의 숫자나 타임스탬프 필드의 텍스트 등 Parquet 파일의 호환되지 않는 필드에 데이터를 저장하려고 하면 null 값이 발생하게 됩니다.

Parquet에 저장하려는 데이터를 로드할 때 기본 동작을 변경할 수 있습니다. 형식을 지정하여 데이터 유형을 변경하거나 태그를 지정하여 Parquet에서 특정 열 유형을 적용할 수 있습니다.

Parquet에 저장할 데이터 형식 지정

Qlik Sense 형식 지정 함수를 사용하여 데이터를 분류할 수 있습니다. 예를 들어, **Text()**, **Num()**, **Interval()** 또는 **Timestamp()**는 데이터를 Parquet에 저장할 때 데이터 형식을 적용할 수 있습니다. Qlik Sense은 필드 특성 및 자동 필드 태그에 따라 거의 20가지 데이터 유형으로 데이터를 저장할 수 있습니다. 자세한 내용은 *해석 함수 (page 1217)*를 참조하십시오.

Num() 및 Text()를 사용하여 데이터 형식 지정

다음 예에서는 Parquet에 저장할 데이터를 준비하는 방법을 보여 줍니다. **Num()**은 숫자 필드에 적용됩니다. **Text()**는 텍스트와 혼합 모두에 적용됩니다. 혼합의 경우 **Text()**는 Parquet의 숫자 필드처럼 처리되어 텍스트 값이 null 값으로 변경되는 것을 방지합니다.

```
Data:
LOAD * INLINE [
num, text, mixed
123.321, abc, 123
456.654, def, xyz
789.987, ghi, 321
];

Format:
NoConcatenate
LOAD num, text, Text(mixed) as mixed RESIDENT Data;
STORE Format INTO [lib://DataFiles/Tmp.parquet] (parquet);
```

Parquet에 저장하기 위한 데이터 태그 지정

Parquet에 데이터를 저장할 때 특정 열 유형을 강제 적용하려면 데이터에 `$parquet` 태그를 지정합니다. 각 데이터 유형은 해당 제어 태그를 추가하여 적용할 수 있습니다. 예를 들어, 필드를 Parquet에 INT32로 저장하려면 로드 스크립트에서 `$parquet-int32` 태그를 지정합니다. 데이터 유형에 따라 이중 데이터의 문자열 또는 숫자 표현이 저장됩니다.

다음 Parquet 제어 태그는 Parquet 파일에 저장하기 위해 필드에 태그를 지정하는 데 사용할 수 있습니다.

Parquet 제어 태그

제어 태그	이중	물리적 유형	논리적 유형	변환된 유형
$parquet-boolean	숫자	BOOLEAN	NONE	NONE
$parquet-int32	숫자	INT32	NONE	NONE
$parquet-int64	숫자	INT64	NONE	NONE
$parquet-float	숫자	FLOAT	NONE	NONE
$parquet-double	숫자	DOUBLE	NONE	NONE
$parquet-bytearray	문자열	BYTE_ARRAY	NONE	UTF8
$parquet-bytearrayfix	숫자	FIXED_LEN_BYTE_ARRAY	NONE	DECIMAL
$parquet-decimal	숫자	INT64	DECIMAL	DECIMAL
$parquet-date	숫자	INT32	DATE	DATE
$parquet-time	숫자	INT64	TIME	TIME_MICROS
$parquet-timestamp	숫자	INT64	TIMESTAMP	TIMESTAMP_MICROS
$parquet-string	문자열	BYTE_ARRAY	STRING	UTF8
$parquet-enum	문자열	BYTE_ARRAY	ENUM	ENUM
$parquet-interval	숫자	FIXED_LEN_BYTE_ARRAY	INTERVAL	INTERVAL
$parquet-json	문자열	BYTE_ARRAY	JSON	JSON
$parquet-bson	문자열	BYTE_ARRAY	BSON	BSON
$parquet-uuid	문자열	FIXED_LEN_BYTE_ARRAY	UUID	NONE

Parquet에 저장하기 위한 데이터 태그 지정

이 예에서는 Parquet에 대한 데이터를 정의하는 데 두 개의 태그가 사용됩니다. *num* 필드에는 $parquet-int32 태그가 지정되어 Parquet에서 INT32로 설정될 숫자 필드로 정의됩니다.

```
Data:
LOAD * INLINE [
num, text,
123.321, abc
456.654, def
789.987, ghi
];
TAG num WITH '$parquet-int32';
STORE Data INTO [lib://DataFiles/Tmp.parquet] (parquet);
```

Table/Tables

Table 및 **Tables** 스크립트 키워드는 **Drop**, **Comment** 및 **Rename** 문과 더불어 **Load** 문의 서식 지정자에서도 사용됩니다.

Tag

이 스크립트 문은 하나 이상의 필드 또는 테이블에 태그를 할당하는 방법을 제공합니다. 앱에 없는 필드 또는 테이블에 태그를 지정하려고 하는 경우 태그는 무시됩니다. 필드 또는 태그 이름의 충돌이 발견되는 경우 마지막 값이 사용됩니다.

구문:

```
Tag [field|fields] fieldlist with tagname
```

```
Tag [field|fields] fieldlist using mapname
```

```
Tag table tablelist with tagname
```

인수

인수	설명
fieldlist	쉼표로 구분된 목록에서 태그를 지정해야 하는 하나 이상의 필드입니다.
mapname	이전에 **mapping Load** 또는 **mapping Select** 문으로 로드한 매핑 테이블의 이름입니다.
tablelist	태그를 지정해야 하는 쉼표로 구분된 테이블 목록입니다.
tagname	필드에 적용할 태그의 이름입니다.

Example 1:

```
tagmap:
mapping LOAD * inline [
a,b
Alpha,MyTag
Num,MyTag
];
tag fields using tagmap;
```

Example 2:

```
tag field Alpha with 'MyTag2';
```

Trace

trace 문이 사용되면 문자열을 **스크립트 실행 진행률** 창과 스크립트 로그 파일에 기록합니다. 이 기능은 디버깅 용도에 매우 유용합니다. **trace** 문보다 먼저 계산되는 변수의 $ 확장을 사용하면 메시지를 사용자 지정할 수 있습니다.

구문:

```
Trace string
```

Example 1:

다음 문은 'Main' 테이블을 로드하는 Load 문 바로 뒤에 사용할 수 있습니다.

```
Trace Main table loaded;
```
그러면 스크립트 실행 대화 상자와 로그 파일에 'Main 테이블이 로드됨' 텍스트가 표시됩니다.

Example 2:

'Main' 테이블을 로드하는 Load 문 바로 뒤에 다음 문을 사용할 수 있습니다.

```
Let MyMessage = NoOfRows('Main') & ' rows in Main table';

Trace $(MyMessage);
```
그러면 스크립트 실행 대화 상자와 로그 파일의 행 수를 나타내는 텍스트가 표시됩니다(예: 'Main 테이블의 265,391행').

Unmap

Unmap 문은 이후에 로드된 필드를 위해 이전의 **Map ... Using** 문을 사용하여 지정된 필드 값 매핑을 비활성화합니다.

구문:

```
Unmap *fieldlist
```

인수:

인수

인수	설명
*fieldlist	스크립트의 이 지점에서 더 이상 매핑하지 않아야 할 필드의 쉼표로 구분된 목록입니다. 필드 목록에 *를 사용하면 모든 필드를 지정할 수 있습니다. 필드 이름에는 와일드카드 문자 * 및 ?가 허용됩니다. 와일드카드를 사용할 경우 필드 이름을 따옴표로 묶어야 합니다.

예 및 결과:

예	결과
Unmap Country;	Country 필드의 매핑을 비활성화합니다.
Unmap A, B, C;	A, B 및 C 필드의 매핑을 비활성화합니다.
Unmap * ;	모든 필드의 매핑을 비활성화합니다.

Unqualify

Unqualify 문은 이전에 **Qualify** 문으로 활성화된 필드 이름의 정규화를 비활성화하는 데 사용됩니다.

구문:

```
Unqualify *fieldlist
```

인수:

인수

인수	설명
*fieldlist	한정을 설정할 필드의 쉼표로 구분된 목록입니다. 필드 목록에 *를 사용하면 모든 필드를 지정할 수 있습니다. 필드 이름에는 와일드카드 문자 * 및 ?가 허용됩니다. 와일드카드를 사용할 경우 필드 이름을 따옴표로 묶어야 합니다. 추가 정보는 **Qualify** 문을 참조하십시오.

Example 1:

친숙하지 않은 데이터베이스에서는 다음 예에 설명된 것과 같이 하나의 필드만 또는 일부 필드만 연결하고 시작하는 것이 유용할 수 있습니다.

```
qualify *;
unqualify TransID;
SQL SELECT * from tab1;
SQL SELECT * from tab2;
SQL SELECT * from tab3;
```

먼저 모든 필드에 대해 한정이 설정됩니다.
다음으로 **TransID**에 대해 한정이 해제됩니다.
tab1, *tab2* 및 *tab3* 테이블 간의 연결에는 **TransID**만 사용됩니다. 다른 모든 필드는 테이블 이름으로 한정됩니다.

Untag

이 스크립트 문은 필드 또는 테이블에서 태그를 제거하는 방법을 제공합니다. 앱에 없는 필드 또는 테이블에서 태그를 제거하려고 하는 경우, 태그 제거는 무시됩니다.

구문:

```
Untag [field|fields] fieldlist with tagname
```

```
Untag [field|fields] fieldlist using mapname
```

```
Untag table tablelist with tagname
```

인수:

인수

인수	설명
fieldlist	쉼표로 구분된 목록에서 태그를 제거해야 하는 하나 이상의 필드입니다.
mapname	이전에 mapping **LOAD** 또는 mapping **SELECT** 문으로 로드한 매핑 테이블의 이름입니다.
tablelist	태그를 지정하지 않아야 하는 쉼표로 구분된 테이블 목록입니다.
tagname	필드에서 제거할 태그의 이름입니다.

Example 1:

```
tagmap:
mapping LOAD * inline [
a,b
Alpha,MyTag
Num,MyTag
];
Untag fields using tagmap;
```

Example 2:

```
Untag field Alpha with MyTag2;
```

3.4 작업 디렉터리

스크립트 문의 파일을 참조하는 경우 경로가 생략되면 Qlik Sense에서 다음 순서로 파일을 검색합니다.

1. **Directory** 문에 의해 지정되는 디렉터리입니다(레거시 스크립트 모드에서만 지원됨).
2. **Directory** 문이 없으면 Qlik Sense가 작업 디렉터리에서 검색합니다.

Qlik Sense Desktop 작업 디렉터리

Qlik Sense Desktop에서 작업 디렉터리는 *C:\Users\{user}\Documents\Qlik\Sense\Apps*입니다.

Qlik Sense 작업 디렉터리

Qlik Sense 서버 설치에서 작업 디렉터리는 Qlik Sense Repository Service에서 지정되며, 기본적으로 *C:\ProgramData\Qlik\Sense\Apps*입니다. 자세한 내용은 Qlik 관리 콘솔 도움말을 참조하십시오.

4 데이터 로드 편집기에서 변수를 사용하여 작업

Qlik Sense의 변수는 숫자 값 또는 영숫자 값 등의 정적 값이나 계산을 저장하는 컨테이너입니다. 앱에서 변수를 사용하면 변수에 대한 변경 내용이 변수가 사용되는 모든 곳에 적용됩니다. 변수는 변수 개요에서 정의하거나 데이터 로드 편집기를 사용하여 스크립트에서 정의할 수 있습니다. 데이터 로드 편집기에서 **Let** 또는 **Set** 문을 사용하여 변수 값을 설정합니다.

시트 편집 시 변수 개요에서 Qlik Sense 변수를 사용하여 작업할 수도 있습니다.

4.1 개요

변수 값의 첫 번째 문자가 등호 '='인 경우 Qlik Sense는 실제 수식 텍스트를 그대로 사용하지 않고 수식(Qlik Sense 표현식)을 값으로 평가한 후 그 결과를 표시하거나 반환합니다.

이를 사용하면 변수가 그 값으로 대체됩니다. 변수는 스크립트에서 달러 기호 확장 및 다양한 제어 문에 사용할 수 있습니다. 이 변수는 스크립트에서 동일한 문자열이 여러 번 반복되는 경우(예: 경로) 매우 유용합니다.

스크립트 실행이 시작될 때 Qlik Sense에서 이전 값에 관계없이 일부 특수 시스템 변수를 설정합니다.

4.2 변수 정의

변수는 정적 값 또는 계산 결과를 저장하는 기능을 제공합니다. 변수를 정의할 때 다음 구문을 사용합니다.

```
set variablename = string
```

또는

```
let variable = expression
```

Set 문은 문자열 할당에 사용됩니다. 등호 오른쪽에 있는 텍스트를 변수에 할당합니다. **Let** 문은 스크립트 런타임에 등호 오른쪽에 있는 표현식을 평가하고 표현식의 결과를 변수에 할당합니다.

변수는 대/소문자가 구분됩니다.

Qlik Sense의 필드 또는 함수와 동일하게 변수 이름을 지정하지 않는 것이 좋습니다.

```
set x = 3 + 4; // 변수는 문자열 '3 + 4'를 값으로 가져옵니다.

let x = 3 + 4; //는 7을 값으로 반환합니다.

set x = Today(); //는 'Today()'를 값으로 반환합니다.

let x = Today(); //는 오늘 날짜를 값으로 반환합니다(예: '9/27/2021').
```

4.3 변수 삭제

스크립트에서 변수를 제거하고 데이터를 다시 로드하면 변수가 앱에서 그대로 유지됩니다. 앱에서 해당 변수를 완전히 제거하려면 변수 대화 상자에서도 삭제해야 합니다.

4.4 변수 값을 필드 값으로 로드

LOAD 문에서 변수 값을 필드 값으로 로드하려는 경우 달러 기호 확장의 결과가 숫자 또는 표현식이 아닌 텍스트이면 확장된 변수를 작은따옴표로 묶어야 합니다.

이 예에서는 스크립트 오류 목록이 포함된 시스템 변수를 테이블에 로드합니다. ScriptErrorList의 확장에는 따옴표가 필요하지만 **If** 절의 ScriptErrorCount의 확장에는 따옴표가 필요 없다는 점에 주의하십시오.

```
IF $(ScriptErrorCount) >= 1 THEN

   LOAD '$(ScriptErrorList)' AS Error AutoGenerate 1;
END IF
```

4.5 변수 계산

Qlik Sense에서 계산된 값이 있는 변수를 사용하는 방법은 여러 가지가 있으며, 결과는 변수 정의 방법과 표현식에서 호출하는 방법에 따라 달라집니다.

이 예에서는 일부 인라인 데이터를 로드합니다.

```
LOAD * INLINE [
    Dim, Sales
    A, 150
    A, 200
    B, 240
    B, 230
    C, 410
    C, 330
];
```

두 개의 변수를 정의하기로 합니다.

```
Let vSales  = 'Sum(Sales)' ;
Let vSales2  = '=Sum(Sales)' ;
```

두 번째 변수에서 표현식 앞에 등호를 추가합니다. 이로서 변수가 확장되고 식이 평가되기 전에 변수가 계산됩니다.

vSales 변수를 그대로 사용하는 경우 측정값 등에서 결과는 문자열 Sum(Sales)로 표시됩니다. 즉, 계산이 수행되지 않습니다.

달러 기호 확장 및 $(vSales) 호출을 표현식에 추가하면 변수가 확장되고 Sales의 합계가 표시됩니다.

마지막으로 $(vSales2)를 호출하면 확장 전에 변수가 계산됩니다. 따라서 표시된 결과는 Sales의 총계입니다. =$(vSales) 및 =$(vSales2)를 측정값 표현식으로 사용했을 때의 차이점은 이 차트에 표시된 결과로 알 수 있습니다.

결과

Dim	$(vSales)	$(vSales2)
A	350	1560
B	470	1560
C	740	1560

표시된 대로 $(vSales)는 차원 값의 부분합이 계산되고 $(vSales2)는 총합이 계산됩니다.

다음과 같은 스크립트 변수를 사용할 수 있습니다.

- *오류 변수 (page 265)*
- *숫자 해석 변수 (page 204)*
- *시스템 변수 (page 196)*
- *값 처리 변수 (page 202)*

4.6 시스템 변수

시스템 변수(일부 시스템 정의 변수)는 시스템 및 Qlik Sense 앱 관련 정보를 제공합니다.

시스템 변수 개요

일부 함수는 개요가 끝난 후에 더 자세히 설명합니다. 이들 함수는 구문에서 함수 이름을 클릭하여 해당 함수에 대한 상세 설명에 즉시 액세스할 수도 있습니다.

CreateSearchIndexOnReload
이 변수는 데이터가 다시 로드되는 동안 검색 인덱스 파일을 만들어야 하는지 여부를 정의합니다.

```
CreateSearchIndexOnReload
```

Floppy
첫 번째로 찾은 플로피 드라이브의 드라이브 문자를 반환하며 일반적으로 *a:*입니다. 이 변수는 시스템 정의 변수입니다.

```
Floppy
```

표준 모드에서는 이 변수가 지원되지 않습니다.

CD
첫 번째로 찾은 CD-ROM 드라이브의 드라이브 문자를 반환합니다. CD-ROM이 발견되지 않으면 *c:*가 반환됩니다. 이 변수는 시스템 정의 변수입니다.

```
CD
```

표준 모드에서는 이 변수가 지원되지 않습니다.

HidePrefix
이 텍스트 문자열로 시작하는 모든 필드 이름은 시스템 필드와 같은 방식으로 숨겨집니다. 이 변수는 사용자 정의 변수입니다.

```
HidePrefix
```

HideSuffix
이 텍스트 문자열로 끝나는 모든 필드 이름은 시스템 필드와 같은 방식으로 숨겨집니다. 이 변수는 사용자 정의 변수입니다.

```
HideSuffix
```

Include
Include/Must_Include 변수는 스크립트에 포함되어 스크립트 코드로 평가되어야 하는 텍스트를 포함한 파일을 지정합니다. 데이터를 추가하는 데 사용되지 않습니다. 스크립트 코드의 일부를 별도의 텍스트 파일에 저장한 후 여러 앱에서 재사용할 수 있습니다. 이 변수는 사용자 정의 변수입니다.

```
$(Include=filename)
$(Must_Include=filename)
```

OpenUrlTimeout
이 변수는 URL 소스에서 데이터를 가져올 때 Qlik Sense에서 지켜야 하는 시간 제한을 초 단위로 정의합니다 (예: HTML 페이지). 이를 생략하면 시간 제한은 약 20분이 됩니다.

```
OpenUrlTimeout
```

QvPath
Qlik Sense 실행 파일에 대한 찾아보기 문자열을 반환합니다. 이 변수는 시스템 정의 변수입니다.

```
QvPath
```

표준 모드에서는 이 변수가 지원되지 않습니다.

QvRoot
Qlik Sense 실행 파일의 루트 디렉터리를 반환합니다. 이 변수는 시스템 정의 변수입니다.

```
QvRoot
```

표준 모드에서는 이 변수가 지원되지 않습니다.

QvWorkPath
현재 Qlik Sense 앱에 대한 탐색 문자열을 반환합니다. 이 변수는 시스템 정의 변수입니다.

```
QvWorkPath
```

표준 모드에서는 이 변수가 지원되지 않습니다.

QvWorkRoot
현재 Qlik Sense 앱의 루트 디렉터리를 반환합니다. 이 변수는 시스템 정의 변수입니다.

```
QvWorkRoot
```

표준 모드에서는 이 변수가 지원되지 않습니다.

StripComments
이 변수를 0으로 설정하면 스크립트에서 /*..*/ 및 // 주석 제거가 금지됩니다. 이 변수를 정의하지 않을 경우 주석 제거가 항상 수행됩니다.

```
StripComments
```

Verbatim
일반적으로 모든 필드 값이 Qlik Sense 데이터베이스에 로드되기 전에 값 앞뒤의 공백 문자(ASCII 32)는 자동으로 제거됩니다. 이 변수를 1로 설정하면 공백 제거가 일시 중단됩니다. 탭(ASCII 9) 및 하드 스페이스(ANSI 160) 문자는 제거되지 않습니다.

```
Verbatim
```

WinPath
Windows에 대한 탐색 문자열을 반환합니다. 이 변수는 시스템 정의 변수입니다.

```
WinPath
```

표준 모드에서는 이 변수가 지원되지 않습니다.

WinRoot
Windows의 루트 디렉터리를 반환합니다. 이 변수는 시스템 정의 변수입니다.

```
WinRoot
```

표준 모드에서는 이 변수가 지원되지 않습니다.

CollationLocale
정렬 순서 및 검색 일치 항목에 사용할 로캘을 지정합니다. 이 값은 로캘의 문화권 이름입니다(예: 'en-US'). 이 변수는 시스템 정의 변수입니다.

```
CollationLocale
```

CreateSearchIndexOnReload

이 변수는 데이터가 다시 로드되는 동안 검색 인덱스 파일을 만들어야 하는지 여부를 정의합니다.

구문:

```
CreateSearchIndexOnReload
```

데이터를 다시 로드하는 동안 검색 인덱스 파일을 만들어야 하는지 또는 사용자의 첫 번째 검색 요청 후 검색 인덱스 파일을 만들어야 하는지 정의할 수 있습니다. 데이터를 다시 로드하는 동안 검색 인덱스 파일을 만들 경우 첫 번째 사용자의 검색 시 발생하는 대기 시간을 방지할 수 있는 이점이 있습니다. 검색 인덱스를 만드는 데 필요한 것보다 더 긴 데이터 다시 로드 시간을 기준으로 가중치를 적용해야 합니다.

이 변수를 생략하면 데이터가 다시 로드되는 동안 검색 인덱스 파일이 만들어지지 않습니다.

세션 앱의 경우 이 변수의 설정에 관계없이 데이터가 다시 로드되는 동안 검색 인덱스 파일이 만들어지지 않습니다.

Example 1: 데이터를 다시 로드하는 동안 검색 인덱스 파일 만들기

```
set CreateSearchIndexOnReload=1;
```

Example 2: 첫 번째 검색 요청 후 검색 인덱스 파일 만들기

```
set CreateSearchIndexOnReload=0;
```

HidePrefix

이 텍스트 문자열로 시작하는 모든 필드 이름은 시스템 필드와 같은 방식으로 숨겨집니다. 이 변수는 사용자 정의 변수입니다.

구문:

```
HidePrefix
```

```
set HidePrefix='_' ;
```

이 문이 사용되면 시스템 필드를 숨길 때 밑줄로 시작하는 필드 이름이 필드 이름 목록에 표시되지 않습니다.

HideSuffix

이 텍스트 문자열로 끝나는 모든 필드 이름은 시스템 필드와 같은 방식으로 숨겨집니다. 이 변수는 사용자 정의 변수입니다.

구문:

```
HideSuffix
```

```
set HideSuffix='%';
```

이 문이 사용되면 시스템 필드를 숨길 때 백분율 기호로 시작하는 필드 이름이 필드 이름 목록에 표시되지 않습니다.

Include

Include/Must_Include 변수는 스크립트에 포함되어 스크립트 코드로 평가되어야 하는 텍스트를 포함한 파일을 지정합니다. 데이터를 추가하는 데 사용되지 않습니다. 스크립트 코드의 일부를 별도의 텍스트 파일에 저장한 후 여러 앱에서 재사용할 수 있습니다. 이 변수는 사용자 정의 변수입니다.

이 변수는 표준 모드의 폴더 데이터 연결만 지원합니다.

구문:

```
$(Include=filename)
```

```
$(Must_Include=filename)
```

두 가지 버전의 변수가 있습니다.

- **Include**는 파일을 찾을 수 없는 경우 오류를 생성하지 않고 아무런 메시지 없이 실패합니다.
- **Must_Include**는 파일을 찾을 수 없으면 오류를 생성합니다.

경로를 지정하지 않으면 Qlik Sense 앱 작업 디렉터리에 상대적인 파일 이름이 사용됩니다. 절대 파일 경로 또는 lib:// 폴더 연결에 대한 경로를 지정할 수도 있습니다. 등호 앞이나 뒤에는 공백 문자를 넣지 않습니다.

set Include =*filename 구조는 적용할 수 없습니다.*

```
$(Include=abc.txt);
```

```
$(Must_Include=lib://DataFiles/abc.txt);
```

제한 사항은

Windows와 Linux에서 UTF-8로 인코딩된 파일 간에 제한된 교차 호환성.

BOM(Byte Order Mark)과 함께 UTF-8을 사용하는 것은 선택 사항입니다. BOM은 비 ASCII 바이트가 파일 시작 부분에 있을 것이라고 예상하지 않은 소프트웨어에서 UTF-8 사용을 방해할 수 있지만, 그렇지 않으면 텍스트 스트림이 처리될 수 있습니다.

- Windows 시스템은 UTF-8의 BOM을 사용하여 바이트 저장소에 모호성이 없음에도 불구하고 파일이 UTF-8로 인코딩되어 있음을 식별합니다.
- Unix/Linux는 유니코드에 UTF-8을 사용하지만, BOM은 명령 파일의 구문을 방해하므로 사용하지 않습니다.

Qlik Sense에 대한 몇 가지 결과가 있습니다.

- Windows에서 UTF-8 BOM으로 시작하는 모든 파일은 UTF-8 스크립트 파일로 간주됩니다. 그렇지 않으면 ANSI 인코딩이 가정됩니다.
- Linux에서 시스템 기본 8비트 코드 페이지는 UTF-8입니다. 따라서 UTF-8은 작동하지만 BOM을 포함하지 않습니다.

결과적으로 이식성을 보장할 수 없습니다. Linux에서 해석할 수 있는 Windows의 파일을 항상 만들 수 있는 것은 아닙니다. 그 반대의 경우도 마찬가지입니다. BOM 처리가 다르기 때문에 UTF-8 인코딩 파일에 대한 두 시스템 간의 교차 호환성이 없습니다.

OpenUrlTimeout

이 변수는 URL 소스에서 데이터를 가져올 때 Qlik Sense에서 지켜야 하는 시간 제한을 초 단위로 정의합니다(예: HTML 페이지). 이를 생략하면 시간 제한은 약 20분이 됩니다.

구문:

```
OpenUrlTimeout
```

```
set OpenUrlTimeout=10;
```

StripComments

이 변수를 0으로 설정하면 스크립트에서 /*..*/ 및 // 주석 제거가 금지됩니다. 이 변수를 정의하지 않을 경우 주석 제거가 항상 수행됩니다.

구문:

```
StripComments
```

특정 데이터베이스 드라이버는 **SELECT** 문에서 최적화 힌트로 /*..*/를 사용합니다. 따라서 이런 경우, **SELECT** 문을 데이터베이스 드라이버로 보내기 전에 주석을 제거해야 합니다.

따라서 필요한 문 바로 다음에 이 변수를 1로 재설정하는 것이 좋습니다.

```
set StripComments=0;
SQL SELECT * /* <optimization directive> */ FROM Table ;
set StripComments=1;
```

Verbatim

일반적으로 모든 필드 값이 Qlik Sense 데이터베이스에 로드되기 전에 값 앞뒤의 공백 문자(ASCII 32)는 자동으로 제거됩니다. 이 변수를 1로 설정하면 공백 제거가 일시 중단됩니다. 탭(ASCII 9) 및 하드 스페이스(ANSI 160) 문자는 제거되지 않습니다.

구문:

```
Verbatim
```

```
set Verbatim = 1;
```

4.7 값 처리 변수

이 섹션에서는 NULL 및 기타 값을 처리하는 데 사용되는 변수에 대해 설명합니다.

값 처리 변수 개요

각 함수는 개요가 끝난 후에 더 자세히 설명합니다. 구문에서 함수 이름을 클릭하여 해당 함수에 대한 상세 설명에 즉시 액세스할 수도 있습니다.

NullDisplay
정의된 기호는 데이터의 최저 수준에서 ODBC 및 커넥터의 모든 NULL 값을 대체합니다. 이 변수는 사용자 정의 변수입니다.

```
NullDisplay
```

NullInterpret
정의된 기호가 텍스트 파일, Excel 파일 또는 인라인 문에서 나올 경우 NULL로 해석됩니다. 이 변수는 사용자 정의 변수입니다.

```
NullInterpret
```

NullValue
NullAsValue 문을 사용하면 지정된 문자열로 지정된 **NullAsValue** 필드의 모든 NULL 값을 정의된 기호로 대체합니다.

```
NullValue
```

OtherSymbol
LOAD/SELECT 문 이전에 '다른 모든 값'으로 취급될 기호를 정의합니다. 이 변수는 사용자 정의 변수입니다.

```
OtherSymbol
```

NullDisplay

정의된 기호는 데이터의 최저 수준에서 ODBC 및 커넥터의 모든 NULL 값을 대체합니다. 이 변수는 사용자 정의 변수입니다.

구문:

```
NullDisplay
```

```
set NullDisplay='<NULL>';
```

NullInterpret

정의된 기호가 텍스트 파일, Excel 파일 또는 인라인 문에서 나올 경우 NULL로 해석됩니다. 이 변수는 사용자 정의 변수입니다.

구문:

```
NullInterpret
```

```
set NullInterpret=' ';
set NullInterpret =;
```

Excel의 빈 값에 대해 NULL 값을 반환하지 않지만 CSV 텍스트 파일의 경우는 NULL 값을 반환합니다.

```
set NullInterpret ='';
```

Excel의 빈 값에 대해 NULL 값을 반환합니다.

NullValue

NullAsValue 문을 사용하면 지정된 문자열로 지정된 **NullAsValue** 필드의 모든 NULL 값을 정의된 기호로 대체합니다.

구문:

```
NullValue
```

```
NullAsValue Field1, Field2;
set NullValue='<NULL>';
```

OtherSymbol

LOAD/SELECT 문 이전에 '다른 모든 값'으로 취급될 기호를 정의합니다. 이 변수는 사용자 정의 변수입니다.

구문:

```
OtherSymbol
```

```
set OtherSymbol='+';
LOAD * inline
```

```
[X, Y
a, a
b, b];
LOAD * inline
[X, Z
a, a
+, c];
```

이제 필드 값 Y='b'는 다른 기호를 통해 Z='c'에 연결됩니다.

4.8 숫자 해석 변수

숫자 해석 변수는 시스템 정의입니다. 변수는 로드 스크립트 상단에 포함되며 스크립트 실행 시 숫자 서식 지정 설정을 적용합니다. 삭제, 편집 또는 복제할 수 있습니다.

숫자 해석 변수는 새로운 앱이 만들어질 때 운영체제의 현재 지역 설정에 따라 자동으로 생성됩니다. Qlik Sense Desktop에서 이는 컴퓨터 운영 체제 설정을 따릅니다. Qlik Sense에서는 Qlik Sense가 설치된 서버의 운영 체제를 따릅니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

통화 서식 지정

MoneyDecimalSep
정의된 소수점 구분 기호는 지역 설정에서 설정한 통화의 소수 기호를 바꿉니다.

```
MoneyDecimalSep
```

MoneyFormat
정의된 기호는 지역 설정에 의해 설정된 통화 기호를 바꿉니다.

```
MoneyFormat
```

MoneyThousandSep
정의된 천 단위 구분 기호는 지역 설정에서 설정한 통화의 숫자 그룹화 기호를 바꿉니다.

```
MoneyThousandSep
```

숫자 서식

DecimalSep
정의된 소수점 구분 기호는 국가별 설정으로 설정된 소수 기호를 바꿉니다.

```
DecimalSep
```

ThousandSep
정의된 천 단위 구분 기호에 따라 운영 체제의 자릿수 구분 기호를 바꿉니다(국가별 설정).

```
ThousandSep
```

NumericalAbbreviation
숫자 약어는 숫자 크기 접두사로 사용할 약어를 설정합니다. 예를 들어 메가 또는 백만의 경우 M(10^6)이고 마이크로의 경우 μ(10^{-6}).

```
NumericalAbbreviation
```

시간 서식 지정

DateFormat
이 환경 변수는 앱에서 기본값으로 사용되는 날짜 형식을 정의합니다. 형식을 사용하여 날짜를 해석하고 해당 형식을 지정합니다. 변수가 정의되지 않은 경우 스크립트가 실행될 때 운영 체제의 국가별 설정 날짜 형식을 가져옵니다.

```
DateFormat
```

TimeFormat
정의된 서식에 따라 운영 체제의 시간 서식을 바꿉니다(국가별 설정).

```
TimeFormat
```

TimestampFormat
정의된 서식에 따라 운영 체제의 날짜 및 시간 서식을 바꿉니다(국가별 설정).

```
TimestampFormat
```

MonthNames
정의된 서식은 국가별 설정의 월 이름 규칙을 바꿉니다.

```
MonthNames
```

LongMonthNames
정의된 서식은 국가별 설정에서 긴 월 이름 규칙을 바꿉니다.

```
LongMonthNames
```

DayNames
정의된 서식은 지역 설정에 의해 설정된 요일 이름 규칙을 바꿉니다.

```
DayNames
```

LongDayNames
정의된 서식은 지역 설정에서 긴 요일 이름 규칙을 바꿉니다.

```
LongDayNames
```

FirstWeekDay
주의 첫 번째 날로 사용할 날짜를 정의하는 정수입니다.

```
FirstWeekDay
```

BrokenWeeks
이 설정은 주를 구분할지 여부를 정의합니다.

BrokenWeeks

ReferenceDay
이 설정은 1주차를 정의하기 위한 기준일로 설정할 1월 날짜를 정의합니다.

ReferenceDay

FirstMonthOfYear
이 설정은 연도의 첫 번째 월로 사용할 월을 정의하며, 월 오프셋을 사용하는 회계 연도를 정의하는 데 사용됩니다(예: 4월 1일에 시작).

이 설정은 현재 사용되지 않지만 향후 사용을 위해 예약되어 있습니다.

유효한 설정은 1(1월) ~ 12(12월)입니다. 기본 설정은 1입니다.

구문:

FirstMonthOfYear

```
Set FirstMonthOfYear=4; //Sets the year to start in April
```

BrokenWeeks

이 설정은 주를 구분할지 여부를 정의합니다.

구문:

BrokenWeeks

Qlik Sense에서는 앱이 만들어질 때 지역 설정을 가져오고 해당 설정은 스크립트에 환경 변수로 저장됩니다.

북미 앱 개발자는 스크립트에서 종종 분리된 주에 해당하는 `Set BrokenWeeks=1;`을 가져옵니다. 유럽 앱 개발자는 스크립트에서 종종 분리되지 않은 주에 해당하는 `Set BrokenWeeks=0;`을 가져옵니다.

분리되지 않은 주는 다음을 의미합니다.

- 어떤 연도에는 1주가 12월에 시작되고 다른 연도에는 전년도의 마지막 주가 1월까지 계속됩니다.
- ISO 8601에 따르면 1주는 항상 1월에 최소 4일이 있습니다. Qlik Sense에서는 `ReferenceDay` 변수를 사용하여 구성할 수 있습니다.

분리된 주는 다음을 의미합니다.

- 연도의 마지막 주가 1월로 이어지지 않습니다.
- 1주차는 1월 1일에 시작되며 대부분의 경우 완전한 한 주가 아닙니다.

다음 값을 사용할 수 있습니다.

- 0(=분리되지 않은 주 사용)
- 1(=분리된 주 사용)

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

주 및 주차에 대해 ISO 설정을 사용하려는 경우 스크립트에 다음이 포함되어 있는지 확인합니다.

```
Set FirstWeekDay=0;
Set BrokenWeeks=0;   //(use unbroken weeks)
Set ReferenceDay=4;
```

미국 설정을 사용하려는 경우 스크립트에 다음이 포함되어 있는지 확인합니다.

```
Set FirstWeekDay=6;
Set BrokenWeeks=1;   //(use broken weeks)
Set ReferenceDay=1;
```

DateFormat

이 환경 변수는 앱에서 기본값으로 사용되는 날짜 서식을 사용하여 date() 및 date#()과 같은 함수를 반환하는 날짜별로 정의합니다. 서식은 날짜를 해석하고 서식을 지정하는 데 사용됩니다. 변수가 정의되지 않은 경우 스크립트가 실행될 때 지역 설정에서 설정한 날짜 서식을 가져옵니다.

구문:

DateFormat

DateFormat 함수 예

예	결과
Set DateFormat='M/D/YY'; //(US format)	DateFormat 함수를 이 방식으로 사용하면 날짜를 미국 서식(월/일/년)으로 정의합니다.
Set DateFormat='DD/MM/YY'; //(UK date format)	DateFormat 함수를 이 방식으로 사용하면 날짜를 영국 서식(일/월/년)으로 정의합니다.
Set DateFormat='YYYY/MM/DD'; //(ISO date format)	DateFormat 함수를 이 방식으로 사용하면 날짜를 ISO 서식(연/월/일)으로 정의합니다.

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

예 1 - 시스템 변수 기본값

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 날짜 데이터 집합.
- 미국 날짜 서식을 사용하는 DateFormat 함수.

이 예에서 데이터 집합은 'Transactions'라는 테이블에 로드됩니다. date 필드가 포함되어 있습니다. 미국 DateFormat 정의가 사용됩니다. 이 패턴은 텍스트 날짜가 로드될 때 암시적 텍스트를 날짜로 변환하는 데 사용됩니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- month

이 측정값을 만듭니다.

```
=sum(amount)
```

결과 테이블

date	월	=sum(amount)
01/01/2022	1월	1000
02/01/2022	2월	2123
03/01/2022	3월	4124
04/01/2022	4월	2431

DateFormat 정의 MM/DD/YYYY는 텍스트를 날짜로 암시적으로 변환하는 데 사용되므로 date 필드가 날짜로 올바르게 해석됩니다. 결과 테이블에 표시된 대로 동일한 형식이 날짜를 표시하는 데 사용됩니다.

예 2 - 시스템 변수 변경

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 이전 예와 동일한 데이터 집합입니다.
- 'DD/MM/YYYY' 형식을 사용하는 DateFormat 함수입니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- month

이 측정값을 만듭니다.

```
=sum(amount)
```

결과 테이블

date	월	=sum(amount)
01/01/2022	1월	1000
02/01/2022	1월	2123
03/01/2022	1월	4124
04/01/2022	1월	2431

DateFormat 정의가 "DD/MM/YYYY"로 설정되었기 때문에 첫 번째 "/" 기호 뒤의 두 자리 숫자가 월로 해석되어 모든 레코드가 1월의 레코드임을 알 수 있습니다.

예 3 – 날짜 해석

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 숫자 형식의 날짜가 있는 데이터 집합.
- 'DD/MM/YYYY' 형식을 사용하는 DateFormat 변수입니다.
- date() 변수.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- month

이 측정값을 만듭니다.

```
=sum(amount)
```

결과 테이블

date	월	=sum(amount)
06/03/2022	6월	1000
06/04/2022	6월	2123
06/05/2022	6월	4124
06/07/2022	6월	2431

로드 스크립트에서 date() 함수를 사용하여 숫자 날짜를 날짜 서식으로 변환합니다. 함수의 두 번째 인수로 지정된 서식을 제공하지 않기 때문에 DateFormat이 사용됩니다. 그 결과 'MM/DD/YYYY' 형식을 사용하는 날짜 필드가 생성됩니다.

예 4 - 외국 날짜 서식

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 날짜 데이터 집합.
- 'DD/MM/YYYY' 서식을 사용하지만 슬래시로 주석 처리되지 않은 DateFormat 변수.

로드 스크립트

```
// 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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- month

이 측정값을 만듭니다.

```
=sum(amount)
```

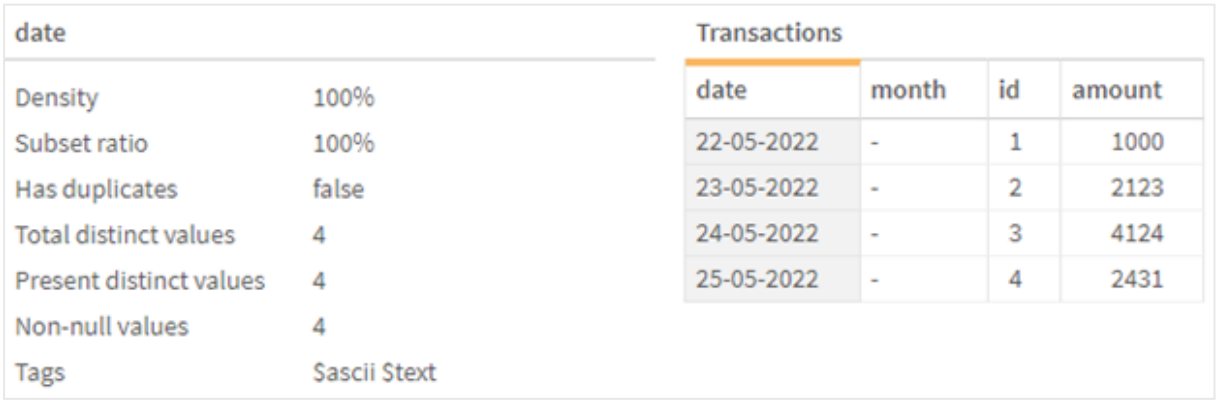
결과 테이블

date	월	=sum(amount)
22-05-2022	-	1000
23-05-2022	-	2123
24-05-2022	-	4124
25-05-2022	-	2431

초기 로드 스크립트에서 사용되는 DateFormat의 기본값은 'MM/DD/YYYY'입니다. 트랜잭션 데이터 집합의 date 필드는 이 형식이 아니기 때문에 필드는 날짜로 해석되지 않습니다. 이는 month 필드 값이 null인 결과 테이블에 표시됩니다.

date 필드의 "Tags" 속성을 검사하여 데이터 모델 뷰어에서 해석된 데이터 유형을 검사할 수 있습니다.

Transactions 테이블의 미리 보기. 텍스트 입력 데이터가 날짜/타임스탬프로 암시적으로 변환되지 않았음을 나타내는 date 필드의 "Tags"에 유의합니다.



이는 DateFormat 시스템 변수를 활성화하여 해결할 수 있습니다.

```
// SET DateFormat='DD/MM/YYYY';
```

이중 슬래시를 제거하고 데이터를 다시 로드합니다.

Transactions 테이블의 미리 보기. 텍스트 입력 데이터가 날짜/타임스탬프로 암시적으로 변환되었음을 나타내는 date 필드의 "Tags"에 유의합니다.

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

정의된 서식은 지역 설정에 의해 설정된 요일 이름 규칙을 바꿉니다.

구문:

DayNames

변수를 수정할 때 개별 값을 구분하기 위해 세미콜론 ; 이 필요합니다.

DayName 함수 예

함수 예	결과 정의
`Set DayNames='Mon;Tue;Wed;Thu;Fri;Sat;Sun';`	DayNames 함수를 이 방식으로 사용하면 축약된 형식으로 요일 이름을 정의합니다.
`Set DayNames='M;Tu;W;Th;F;Sa;Su';`	DayNames 함수를 이 방식으로 사용하면 첫 문자로 요일 이름을 정의합니다.

DayNames 함수는 다음 함수와 함께 사용되는 경우가 많습니다.

관련 함수

함수	상호 작용
weekday (page 1030)	DayNames 를 필드 값으로 반환하는 스크립트 함수.
Date (page 1187)	DayNames 를 필드 값으로 반환하는 스크립트 함수.
LongDayNames (page 224)	DayNames의 긴 형식 값.

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

예 1 - 시스템 변수 기본값

로드 스크립트 및 결과

개요

이 예에서 데이터 집합의 날짜는 MM/DD/YYYY 서식으로 설정됩니다.

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Transactions라는 테이블에 로드되는 날짜 데이터 집합.
- date 필드.
- 기본 DayNames 정의.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- dayname

이 측정값을 만듭니다.

```
sum(amount)
```

결과 테이블

date	dayname	sum(amount)
01/01/2022	Sat	1000
02/01/2022	Tue	2123
03/01/2022	Tue	4124
04/01/2022	Fri	2431

이 로드 스크립트에서 WeekDay 함수는 제공된 인수로 date 필드와 함께 사용됩니다. 결과 테이블에서 이 WeekDay 함수의 출력은 DayNames 정의 서식으로 요일을 표시합니다.

예 2 - 시스템 변수 변경

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다. 첫 번째 예와 동일한 데이터 집합과 시나리오가 사용됩니다.

그러나 스크립트 시작 시 DayNames 정의를 수정하여 Afrikaans에서는 요일을 축약하여 사용됩니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- dayname

이 측정값을 만듭니다.

```
sum(amount)
```

결과 테이블

date	dayname	sum(amount)
01/01/2022	Sa	1000
02/01/2022	Di	2123
03/01/2022	Di	4124
04/01/2022	Vr	2431

결과 테이블에서 이 WeekDay 함수의 출력은 DayNames 정의 서식으로 요일을 표시합니다.

DayNames에 대한 언어가 이 예에서와 같이 수정된 경우에도 LongDayNames에 여전히 영어로 요일이 포함된다는 점을 기억해야 합니다. 응용 프로그램에서 두 변수를 모두 사용하는 경우에도 수정해야 합니다.

예 3 – 날짜 함수

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Transactions라는 테이블에 로드되는 날짜 데이터 집합.
- date 필드.
- 기본 DayNames 정의.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- dayname

이 측정값을 만듭니다.

```
sum(amount)
```

결과 테이블

date	dayname	sum(amount)
01/01/2022	Sat	1000
02/01/2022	Tue	2123
03/01/2022	Tue	4124
04/01/2022	Fri	2431

기본 DayNames 정의가 사용됩니다. 이 로드 스크립트에서 Date 함수는 첫 번째 인수로 date 필드와 함께 사용됩니다. 두 번째 인수는 WWW입니다. 이 서식 지정은 결과를 DayNames 정의에 저장된 값으로 변환합니다. 이는 결과 테이블의 출력에 표시됩니다.

DecimalSep

정의된 소수점 구분 기호는 국가별 설정으로 설정된 소수 기호를 바꿉니다.

Qlik Sense는 인식 가능한 숫자 패턴이 나타날 때마다 자동으로 텍스트를 숫자로 해석합니다. ThousandSep 및 DecimalSep 시스템 변수는 텍스트를 숫자로 구문 분석할 때 적용되는 패턴의 구성을 결정합니다. ThousandSep 및 DecimalSep 변수는 프런트 엔드 차트 및 테이블에서 숫자 콘텐츠를 시각화할 때 기본 숫자 서식 패턴을 설정합니다. 즉, 프런트 엔드 표현식의 **숫자 서식 지정** 옵션에 직접적인 영향을 줍니다.

쉼표 ','의 천 단위 구분 기호와 '.'의 소수점 구분 기호를 가정하면 다음은 암시적으로 동등한 숫자 값으로 변환되는 패턴의 예입니다.

```
0,000.00
```

```
0000.00
```

```
0,000
```

다음은 텍스트로 변경되지 않은 상태로 유지되는 패턴의 예입니다. 즉, 숫자로 변환되지 않습니다.

```
0.000,00
```

```
0,00
```

구문:

DecimalSep

함수 예

예	결과
`Set DecimalSep='.';`	소수점 구분 기호로 '.'를 설정합니다.
`Set DecimalSep=',';`	소수점 구분 기호로 ','를 설정합니다.

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 `SET DateFormat` 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

예 – 다른 입력 데이터에 대한 숫자 구분 기호 변수 설정의 영향

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 다른 서식 패턴으로 설정된 합계가 있는 합계 및 날짜의 데이터 집합.
- `Transactions`라는 테이블.
- '.'로 설정된 `DecimalSep` 변수.
- ','로 설정된 `ThousandSep` 변수.
- '|' 문자로 설정된 `delimiter` 변수. 한 줄에서 다른 필드를 구분하는 데 사용됩니다.

로드 스크립트

```
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 '|');
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원 amount로 추가합니다.

이 측정값을 만듭니다.

```
=sum(amount)
```

결과 테이블

Amount	**=Sum(amount)**
Totals	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

숫자로 해석되지 않는 모든 값은 텍스트로 남아 있으며 기본적으로 왼쪽에 정렬됩니다. 성공적으로 변환된 값은 원래 입력 서식을 유지하면서 오른쪽에 정렬됩니다.

표현식 열은 기본적으로 소수점 구분 기호 '.'로만 서식이 지정된 동등한 숫자를 표시합니다. 표현식 구성에서 **숫자 서식 지정** 드롭다운 설정으로 재정의할 수 있습니다.

FirstWeekDay

주의 첫 번째 날로 사용할 날짜를 정의하는 정수입니다.

구문:

FirstWeekDay

월요일은 날짜 및 시간 표시에 대한 국제 표준인 ISO 8601에 따라 주의 첫 번째 요일입니다. 또한 월요일은 영국, 프랑스, 독일, 스웨덴과 같은 여러 국가에서 주의 첫 번째 요일로 사용됩니다.

그러나 미국과 캐나다와 같은 다른 국가에서는 일요일이 주의 시작으로 간주됩니다.

Qlik Sense에서는 앱이 만들어질 때 지역 설정을 가져오고 해당 설정은 스크립트에 환경 변수로 저장됩니다.

북미 앱 개발자는 스크립트에서 종종 일요일에 해당하는 `Set FirstWeekDay=6;`을 가져옵니다. 유럽 앱 개발자는 스크립트에서 종종 월요일에 해당하는 `Set FirstWeekDay=0;`을 가져옵니다.

FirstWeekDay에 설정할 수 있는 값

값	일
0	월요일
1	화요일
2	수요일
3	목요일
4	금요일
5	토요일
6	일요일

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 `SET DateFormat` 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

주 및 주차에 대해 ISO 설정을 사용하려는 경우 스크립트에 다음이 포함되어 있는지 확인합니다.

```
Set FirstWeekDay=0;  // Monday as first week day
Set BrokenWeeks=0;
Set ReferenceDay=4;
```

미국 설정을 사용하려는 경우 스크립트에 다음이 포함되어 있는지 확인합니다.

```
Set FirstWeekDay=6;  // Sunday as first week day
Set BrokenWeeks=1;
Set ReferenceDay=1;
```

예 1 – 기본값 사용(스크립트)

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

이 예에서 로드 스크립트는 기본 Qlik Sense 시스템 변수 값인 `FirstWeekDay=6`을 사용합니다. 이 데이터에는 2020년의 처음 14일 동안의 데이터가 포함되어 있습니다.

로드 스크립트

```
// 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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- week
- weekday

결과 테이블

Date	주	weekday
01/01/2021	1	Wed
01/02/2021	1	Thu
01/03/2021	1	Fri
01/04/2021	1	Sat
01/05/2021	2	Sun
01/06/2020	2	Mon
01/07/2020	2	Tue
01/08/2020	2	Wed
01/09/2020	2	Thu
01/10/2020	2	Fri
01/11/2020	2	Sat
01/12/2020	3	Sun
01/13/2020	3	Mon
01/14/2020	3	Tue

기본 설정이 사용 중이므로 FirstWeekDay 시스템 변수는 6으로 설정됩니다. 결과 테이블에서 각각의 새로운 주는 일요일(1월 5일 및 12일)에 시작한다는 것을 알 수 있습니다.

예 2 – FirstWeekDay 변수 변경(스크립트)

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

이 예에서 데이터에는 2020년의 처음 14일이 포함됩니다. 스크립트 시작 시 FirstWeekDay 변수를 3으로 설정합니다.

로드 스크립트

```
// 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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- week
- weekday

결과 테이블

Date	주	weekday
01/01/2021	52	Wed
01/02/2021	1	Thu
01/03/2021	1	Fri
01/04/2021	1	Sat
01/05/2021	1	Sun
01/06/2020	1	Mon
01/07/2020	1	Tue
01/08/2020	1	Wed
01/09/2020	2	Thu
01/10/2020	2	Fri
01/11/2020	2	Sat
01/12/2020	2	Sun
01/13/2020	2	Mon
01/14/2020	2	Tue

FirstWeekDay 시스템 변수가 3으로 설정되어 있으므로 각 주의 첫째 날은 목요일이 됩니다. 결과 테이블에서 각각의 새로운 주가 목요일(1월 2일 및 9일)에 시작한다는 것을 알 수 있습니다.

LongDayNames

정의된 서식은 지역 설정에서 긴 요일 이름 규칙을 바꿉니다.

구문:

```
LongDayNames
```

LongDayNames 함수의 다음 예는 전체 요일 이름을 정의합니다.

```
Set LongDayNames='Monday;Tuesday;Wednesday;Thursday;Friday;Saturday;Sunday';
```

변수를 수정할 때 개별 값을 구분하기 위해 세미콜론 ;이 필요합니다.

LongDayNames 함수는 DayNames를 필드 값으로 반환하는 *Date (page 1187)* 함수와 함께 사용할 수 있습니다.

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

예 1 - 시스템 변수 기본값

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Transactions라는 테이블에 로드되는 날짜 데이터 집합.
- date 필드.
- 기본 LongDayNames 정의.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- dayname

이 측정값을 만듭니다.

```
=sum(amount)
```

결과 테이블

date	dayname	=sum(amount)
01/01/2022	토요일	1000
02/01/2022	화요일	2123
03/01/2022	화요일	4124
04/01/2022	금요일	2431

이 로드 스크립트에서 dayname이라는 필드를 만들기 위해 Date 함수는 첫 번째 인수로 date 필드와 함께 사용됩니다. 이 함수의 두 번째 인수는 서식 지정 wwww입니다.

이 서식 지정을 사용하면 첫 번째 인수의 값이 변수 LongDayNames에 설정된 해당하는 전체 요일 이름으로 변환됩니다. 결과 테이블에서 만들어진 필드 dayname의 필드 값이 이를 표시합니다.

예 2 - 시스템 변수 변경

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

첫 번째 예와 동일한 데이터 집합과 시나리오가 사용됩니다. 그러나 스크립트 시작 시 스페인어로 요일을 사용하도록 LongDayNames 정의가 수정됩니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- dayname

이 측정값을 만듭니다.

```
=sum(amount)
```

결과 테이블

date	dayname	=sum(amount)
01/01/2022	Sábado	1000
02/01/2022	Martes	2123
03/01/2022	Martes	4124
04/01/2022	Viernes	2431

이 로드 스크립트에서 LongDayNames 변수는 스페인어로 요일을 나열하도록 수정됩니다.

그런 다음 dayname이라는 필드를 만들며, 이는 첫 번째 인수로 date 필드와 함께 사용되는 Date 함수입니다.

이 함수의 두 번째 인수는 서식 지정 WWWW입니다. 이 서식 지정을 사용하면 Qlik Sense에서 첫 번째 인수의 값이 변수 LongDayNames에 설정된 해당하는 전체 요일 이름으로 변환됩니다.

결과 테이블에서 만들어진 필드 dayname의 필드 값은 스페인어로 전체 요일을 표시합니다.

LongMonthNames

정의된 서식은 국가별 설정에서 긴 월 이름 규칙을 바꿉니다.

구문:

```
LongMonthNames
```

이 변수를 수정할 때 개별 값을 구분하려면 ;을 사용해야 합니다.

LongMonthNames 함수의 다음 예는 전체 월 이름을 정의합니다.

```
Set
LongMonthNames='January;February;March;April;May;June;July;August;September;October;November;D
ecember';
```

LongMonthNames 함수는 다음 함수와 함께 사용되는 경우가 많습니다.

관련 함수

함수	상호 작용
Date (page 1187)	DayNames를 필드 값으로 반환하는 스크립트 함수.
LongDayNames (page 224)	DayNames의 긴 형식 값.

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

예 1 - 시스템 변수 기본값

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Transactions라는 테이블에 로드되는 날짜 데이터 집합.
- date 필드.
- 기본 LongMonthNames 정의.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- monthname

이 측정값을 만듭니다.

```
=sum(amount)
```

결과 테이블

date	monthname	sum(amount)
01/01/2022	1월	1000.45
01/02/2022	1월	2123.34
01/03/2022	1월	4124.35
01/04/2022	1월	2431.36
01/05/2022	1월	4787.78
01/06/2022	1월	2431.84
01/07/2022	1월	2854.83
01/08/2022	1월	3554.28
01/09/2022	1월	3756.17
01/10/2022	1월	3454.35

기본 LongMonthNames 정의가 사용됩니다. 이 로드 스크립트에서 month이라는 필드를 만들기 위해 Date 함수는 첫 번째 인수로 date 필드와 함께 사용됩니다. 이 함수의 두 번째 인수는 서식 지정 MMMM입니다.

이 서식 지정을 사용하면 Qlik Sense에서 첫 번째 인수의 값이 변수 LongMonthNames에 설정된 해당하는 전체 월 이름으로 변환됩니다. 결과 테이블에서 만들어진 필드 month의 필드 값이 이를 표시합니다.

예 2 - 시스템 변수 변경

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Transactions라는 테이블에 로드되는 날짜 데이터 집합.
- date 필드.
- 스페인어로 축약된 요일을 사용하도록 수정된 LongMonthNames 변수.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고, 측정값으로 sum(amount)를 추가하고, 다음 필드를 차원으로 추가합니다.

- date
- monthname

이 측정값을 만듭니다.

```
=sum(amount)
```

결과 테이블

date	monthname	sum(amount)
01/01/2022	Enero	1000.45
01/02/2022	Enero	2123.34
01/03/2022	Enero	4124.35

date	monthname	sum(amount)
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

이 로드 스크립트에서 LongMonthNames 변수는 스페인어로 연도의 월을 나열하도록 수정됩니다. 그런 다음 monthname이라는 필드를 만들기 위해 Date 함수가 첫 번째 인수로 date 필드와 함께 사용됩니다. 이 함수의 두 번째 인수는 서식 지정 MMMM입니다.

이 서식 지정을 사용하면 Qlik Sense에서 첫 번째 인수의 값이 변수 LongMonthNames에 설정된 해당하는 전체 월 이름으로 변환됩니다. 결과 테이블에서 만들어진 필드 monthname 의 필드 값은 스페인어로 월 이름을 표시합니다.

MoneyDecimalSep

정의된 소수점 구분 기호는 지역 설정에서 설정한 통화의 소수 기호를 바꿉니다.

> *기본적으로 Qlik Sense는 테이블 차트에서 숫자와 텍스트를 다르게 표시합니다. 숫자는 오른쪽으로 정렬되고 텍스트는 왼쪽으로 정렬됩니다. 이렇게 하면 텍스트에서 숫자로의 변환 문제를 쉽게 찾을 수 있습니다. Qlik Sense 결과를 표시하는 이 페이지의 모든 테이블은 이 형식을 사용합니다.*

구문:

```
MoneyDecimalSep
```

Qlik Sense 응용 프로그램은 이 서식 지정을 따르는 텍스트 필드를 화폐 값으로 해석합니다. 텍스트 필드에는 MoneyFormat 시스템 변수에 정의된 통화 기호가 포함되어야 합니다. MoneyDecimalSep는 여러 다른 지역 설정에서 받은 데이터 소스를 처리할 때 특히 유용합니다.

다음 예에서는 MoneyDecimalSep 시스템 변수의 사용 사례를 보여 줍니다.

```
Set MoneyDecimalSep='.';
```

이 함수는 종종 다음 함수와 함께 사용됩니다.

관련 함수

함수	상호 작용
MoneyFormat	텍스트 필드 해석의 경우 MoneyFormat 기호가 해석의 일부로 사용됩니다. 숫자 서식 지정의 경우 Qlik Sense는 차트 개체에서 MoneyFormat 서식 지정을 사용합니다.
MoneyThousandSep	텍스트 필드 해석의 경우 MoneyThousandSep 함수도 준수해야 합니다.

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

예 1 - MoneyDecimalSep 점(.) 표기법

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Transactions라는 테이블에 로드되는 데이터 집합.
- 소수점 구분 기호로 점 '.'을 사용하는 텍스트 서식의 통화 필드가 있는 제공된 데이터. '£' 기호가 접두사로 붙는 마지막 레코드를 제외하고 각 레코드에는 '$' 기호가 접두사로 붙습니다.

MoneyFormat 시스템 변수는 달러 '$'를 기본 통화로 정의합니다.

로드 스크립트

```
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'
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다.amount.

다음 측정값을 추가합니다.

- isNum(amount)
- sum(amount)

아래 결과를 검토하여 모든 달러 '$' 값의 올바른 해석을 보여 줍니다.

결과 테이블

amount	=isNum(amount)	=Sum(amount)
Totals	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

위의 결과 테이블은 달러($)가 접두사로 붙은 모든 값에 대해 amount 필드가 올바르게 해석되지만 파운드(£)가 접두사로 붙은 amount는 금전적 가치로 변환되지 않은 방법을 보여 줍니다.

예 2 - MoneyDecimalSep 쉼표(,) 표기법

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Transactions라는 테이블에 로드되는 데이터 집합.
- 소수점 구분 기호로 쉼표 ','를 사용하는 텍스트 서식의 통화 필드가 있는 제공된 데이터. 소수점 구분 기호 '.'을 잘못 사용하는 마지막 레코드를 제외하고 각 레코드에는 '$' 기호가 접두사로 붙습니다.

MoneyFormat 시스템 변수는 달러 '$'를 기본 통화로 정의합니다.

로드 스크립트

```
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'
];
```

결과

결과에 대한 단락 텍스트입니다.

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다.amount.

다음 측정값을 추가합니다.

- isNum(amount)
- sum(amount)

아래 결과를 검토합니다. 소수점 구분 기호 점 '.' 표기법을 사용하는 금액을 제외하고 모든 값이 올바르게 해석됨을 볼 수 있습니다. 이 경우 쉼표를 대신 사용해야 합니다.

결과 테이블

amount	=isNum(amount)	=Sum(amount)
Totals	0	$3905.98

amount	=isNum(amount)	=Sum(amount)
$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

이 시스템 변수는 숫자에 화폐 기호가 접두사로 붙은 텍스트를 숫자로 자동 변환하기 위해 Qlik이 사용하는 서식 패턴을 정의합니다. 또한 숫자 서식 지정 속성이 '화폐'로 설정된 측정값이 차트 개체에 표시되는 방식을 정의합니다.

MoneyFormat 시스템 변수에서 서식 패턴의 일부로 정의된 기호는 국가별 설정으로 설정된 통화 기호를 바꿉니다.

기본적으로 Qlik Sense는 테이블 차트에서 숫자와 텍스트를 다르게 표시합니다. 숫자는 오른쪽으로 정렬되고 텍스트는 왼쪽으로 정렬됩니다. 이렇게 하면 텍스트에서 숫자로의 변환 문제를 쉽게 찾을 수 있습니다. Qlik Sense 결과를 표시하는 이 페이지의 모든 테이블은 이 형식을 사용합니다.

구문:

MoneyFormat

```
Set MoneyFormat='$ #,##0.00; ($ #,##0.00)';
```

이 서식 지정은 숫자 필드의 Number Formatting 속성이 Money로 설정된 경우 차트 개체에 표시됩니다. 또한 숫자 텍스트 필드를 Qlik Sense에서 해석할 때 텍스트 필드의 통화 기호가 MoneyFormat 변수에 정의된 기호와 일치하면 Qlik Sense는 이 필드를 화폐 값으로 해석합니다.

이 함수는 종종 다음 함수와 함께 사용됩니다.

관련 함수

함수	상호 작용
MoneyDecimalSep (page 230)	숫자 서식 지정의 경우 MoneyDecimalSep는 개체의 필드 서식 지정에 사용됩니다.
MoneyThousandSep (page 238)	숫자 서식 지정의 경우 MoneyThousandSep는 개체의 필드 서식 지정에 사용됩니다.

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

예 1 - MoneyFormat

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

이 로드 스크립트에는 Transactions라는 테이블에 로드되는 데이터 집합이 포함되어 있습니다. 기본 MoneyFormat 변수 정의가 사용됩니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- amount

이 측정값을 추가합니다.

```
=Sum(amount)
```

숫자 서식 지정에서 **통화**를 선택하여 Sum(amount)를 통화 값으로 구성합니다.

결과 테이블

date	Amount	=Sum(amount)
Totals		$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

기본 MoneyFormat 정의가 사용됩니다. 이는 다음과 같이 보입니다. $###0.00;-$###0.00. 결과 테이블에서 amount 필드 서식으로 통화 기호가 표시되며 소수점과 소수 자릿수가 포함되었습니다.

예 2 - 천 단위 구분 기호 및 혼합 입력 서식이 있는 MoneyFormat

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Transactions라는 테이블에 로드되는 혼합 입력 서식 데이터 집합. 천 단위 구분 기호와 소수점 구분 기호가 배치되어 있습니다.
- MoneyFormat 정의를 수정하면 천 단위 구분 기호로 쉼표를 포함하도록 수정됩니다.
- 데이터 행 중 하나가 잘못된 위치에서 천 단위 구분 기호 쉼표로 잘못 구분되었습니다. 이 금액이 텍스트로 남아 있고 숫자로 해석할 수 없는 방법에 유의합니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- amount

이 측정값을 추가합니다.

```
=Sum(amount)
```

숫자 서식 지정에서 **통화**를 선택하여 Sum(amount)를 통화 값으로 구성합니다.

결과 테이블

date	Amount	=Sum(amount)
Totals		$119,548,811,911.90

date	Amount	=Sum(amount)
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

스크립트 시작 시 MoneyFormat 시스템 변수가 천 단위 구분 기호로 쉼표를 포함하도록 수정됩니다. Qlik Sense 테이블에서 이 구분 기호를 포함하는 서식 지정을 볼 수 있습니다. 또한 잘못된 구분 기호가 있는 행이 올바르게 해석되지 않고 텍스트로 남아 있습니다. 따라서 금액 합계에 적용되지 않습니다.

MoneyThousandSep

정의된 천 단위 구분 기호는 지역 설정에서 설정한 통화의 숫자 그룹화 기호를 바꿉니다.

기본적으로 Qlik Sense는 테이블 차트에서 숫자와 텍스트를 다르게 표시합니다. 숫자는 오른쪽으로 정렬되고 텍스트는 왼쪽으로 정렬됩니다. 이렇게 하면 텍스트에서 숫자로의 변환 문제를 쉽게 찾을 수 있습니다. Qlik Sense 결과를 표시하는 이 페이지의 모든 테이블은 이 형식을 사용합니다.

구문:

MoneyThousandSep

Qlik Sense 응용 프로그램은 이 서식 지정을 따르는 텍스트 필드를 화폐 값으로 해석합니다. 텍스트 필드에는 MoneyFormat 시스템 변수에 정의된 통화 기호가 포함되어야 합니다. MoneyThousandSep는 여러 다른 지역 설정에서 받은 데이터 소스를 처리할 때 특히 유용합니다.

다음 예에서는 MoneyThousandSep 시스템 변수의 사용 사례를 보여 줍니다.

```
Set MoneyDecimalSep=',';
```

이 함수는 종종 다음 함수와 함께 사용됩니다.

관련 함수

함수	상호 작용
MoneyFormat	텍스트 필드 해석의 경우 MoneyFormat 기호가 해석의 일부로 사용됩니다. 숫자 서식 지정의 경우 MoneyFormat는 차트 개체에서 Qlik Sense 서식 지정을 사용합니다.
MoneyDecimalSep	텍스트 필드 해석의 경우 MoneyDecimalSep 함수도 준수해야 합니다.

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

예 1 - MoneyThousandSep 쉼표(,) 표기법

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Transactions라는 테이블에 로드되는 데이터 집합.
- 제공된 데이터에 천 단위 구분 기호로 쉼표 ','를 사용하는 텍스트 서식의 통화 필드가 있습니다. 또한 각 레코드에는 '$' 기호가 접두사로 붙습니다.

MoneyFormat 시스템 변수는 달러 '$'를 기본 통화로 정의합니다.

로드 스크립트

```
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'
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다.amount.

다음 측정값을 추가합니다.

- isNum(amount)
- sum(amount)

아래 결과를 검토합니다. 이 테이블에서는 천 단위 구분 기호로 쉼표 ',' 표기법을 사용하여 모든 값의 올바른 해석을 보여 줍니다.

amount 필드는 천 단위 구분 기호로 점 '.'을 사용하는 하나의 값을 제외하고 모든 값에 대해 올바르게 해석되었습니다.

결과 테이블

amount	=isNum(amount)	=Sum(amount)
Totals	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

예 2 - MoneyThousandSep 점(.) 표기법

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Transactions라는 테이블에 로드되는 데이터 집합.
- 천 단위 구분 기호로 점 '.'을 사용하는 텍스트 서식의 통화 필드가 있는 제공된 데이터. 또한 각 레코드에는 '$' 기호가 접두사로 붙습니다.

MoneyFormat 시스템 변수는 달러 '$'를 기본 통화로 정의합니다.

로드 스크립트

```
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'
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다.amount.

다음 측정값을 추가합니다.

- isNum(amount)
- sum(amount)

아래 결과를 검토합니다. 천 단위 구분 기호로 점 '.' 표기법을 사용하여 모든 값의 올바른 해석을 보여 줍니다.

amount 필드는 천 단위 구분 기호로 쉼표 ','를 사용하는 하나의 값을 제외하고 모든 값에 대해 올바르게 해석되었습니다.

결과 테이블

amount	=isNum(amount)	=Sum(amount)
Totals	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

정의된 서식은 국가별 설정의 월 이름 규칙을 바꿉니다.

구문:

```
MonthNames
```

이 변수를 수정할 때 개별 값을 구분하려면 ;을 사용해야 합니다.

함수 예

예	결과
`Set MonthNames='Jan;Feb;Mar;Apr;May;Jun;Jul;Aug;Sep;Oct;Nov;Dec';`	MonthNames 함수를 이 방식으로 사용하면 영어로 된 월 이름과 약어 형식이 정의됩니다.
`Set MonthNames='Enero;Feb;Marzo;Abr;Mayo;Jun;Jul;Agosto;Set;Oct;Nov;Dic';`	MonthNames 함수를 이 방식으로 사용하면 스페인어로 된 월 이름과 약어 형식이 정의됩니다.

MonthNames 함수는 다음 함수와 함께 사용할 수 있습니다.

관련 함수

함수	상호 작용
month (page 877)	MonthNames에 필드 값으로 정의된 값을 반환하는 스크립트 함수
Date (page 1187)	제공된 서식 지정 인수를 기반으로 MonthNames에 필드 값으로 정의된 값을 반환하는 스크립트 함수
LongMonthNames (page 226)	MonthNames의 긴 형식 값

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

예 1 - 시스템 변수 기본값

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Transactions라는 테이블에 로드되는 날짜 데이터 집합.
- date 필드.
- 기본 MonthNames 정의.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- monthname

이 측정값을 만듭니다.

```
=sum(amount)
```

결과 테이블

date	monthname	sum(amount)
01/01/2022	1월	1000.45
01/02/2022	1월	2123.34
01/03/2022	1월	4124.35
01/04/2022	1월	2431.36
01/05/2022	1월	4787.78
01/06/2022	1월	2431.84
01/07/2022	1월	2854.83
01/08/2022	1월	3554.28
01/09/2022	1월	3756.17
01/10/2022	1월	3454.35

기본 MonthNames 정의가 사용됩니다. 이 로드 스크립트에서 Month 함수는 제공된 인수로 date 필드와 함께 사용됩니다.

결과 테이블에서 이 Month 함수의 출력은 MonthNames 정의 서식으로 연도의 월을 표시합니다.

예 2 - 시스템 변수 변경

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Transactions라는 테이블에 로드되는 날짜 데이터 집합.
- date 필드.
- 스페인어로 축약된 월을 사용하도록 수정된 MonthNames 변수.

로드 스크립트

```
                              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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- monthname

이 측정값을 만듭니다.

```
=sum(amount)
```

결과 테이블

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

date	monthname	sum(amount)
01/08/2022	Enero	3554.28
01/09/2022	Enero	3756.17
01/10/2022	Enero	3454.35

이 로드 스크립트에서는 먼저 MonthNames 변수가 스페인어로 축약된 연도의 월을 나열하도록 수정됩니다. Month 함수는 제공된 인수로 date 필드와 함께 사용됩니다.

결과 테이블에서 이 Month 함수의 출력은 MonthNames 정의 서식으로 연도의 월을 표시합니다.

MonthNames 변수에 대한 언어가 이 예에서와 같이 수정된 경우에도 LongMonthNames 변수에 여전히 영어로 월이 포함된다는 점을 기억해야 합니다. 응용 프로그램에서 두 변수를 모두 사용하는 경우 LongMonthNames 변수를 수정해야 합니다.

예 3 – 날짜 함수

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Transactions라는 테이블에 로드되는 날짜 데이터 집합.
- date 필드.
- 기본 MonthNames 정의.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- monthname

이 측정값을 만듭니다.

```
=sum(amount)
```

결과 테이블

date	monthname	sum(amount)
01/01/2022	1월	1000.45
01/02/2022	1월	2123.34
01/03/2022	1월	4124.35
01/04/2022	1월	2431.36
01/05/2022	1월	4787.78
01/06/2022	1월	2431.84
01/07/2022	1월	2854.83
01/08/2022	1월	3554.28
01/09/2022	1월	3756.17
01/10/2022	1월	3454.35

기본 MonthNames 정의가 사용됩니다. 이 로드 스크립트에서 Date 함수는 첫 번째 인수로 date 필드와 함께 사용됩니다. 두 번째 인수는 MMM입니다.

이 서식 지정을 사용하면 Qlik Sense에서 첫 번째 인수의 값이 변수 MonthNames에 설정된 해당하는 월 이름으로 변환됩니다. 결과 테이블에서 만들어진 필드 month의 필드 값이 이를 표시합니다.

NumericalAbbreviation

숫자 약어는 숫자 크기 접두사로 사용할 약어를 설정합니다. 예를 들어 메가 또는 백만의 경우 M(10^6)이고 마이크로의 경우 μ(10^{-6}).

구문:

NumericalAbbreviation

NumericalAbbreviation 변수를 세미콜론으로 구분된 약어 정의 쌍 목록을 포함하는 문자열로 설정합니다. 각 약어 정의 쌍에는 소수 자릿수(십진수의 지수)와 콜론으로 구분된 약어(예: 백만의 경우 6:M)가 포함되어야 합니다.

기본 설정은 '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' 입니다.

이 설정은 천의 접두사를 t로 바꾸고, 10억의 접두사를 B로 변경합니다. 이는 t$, M$ 및 B$와 같은 약어가 필요한 금융 응용 프로그램에서 유용합니다.

```
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

이 설정을 통해 1주차를 정의하기 위해 기준일로 설정할 1월의 날짜를 정의합니다. 즉, 이 설정으로 1주차의 며칠이 1월 내의 날짜여야 하는지 지정합니다.

구문:

ReferenceDay

ReferenceDay는 해당 연도의 첫 번째 주에 며칠이 포함되는지 설정합니다. ReferenceDay는 1과 7 사이의 값으로 설정할 수 있습니다. 1-7 범위를 벗어난 값은 주의 중간(4)으로 해석됩니다. 즉, ReferenceDay가 4로 설정되는 것과 같습니다.

ReferenceDay 설정 값을 선택하지 않은 경우 기본값은 아래 ReferenceDay 값 표에서 볼 수 있듯이 주의 중간(4)으로 해석되는 ReferenceDay=0을 보여 줍니다.

ReferenceDay 함수는 다음 함수와 함께 사용되는 경우가 많습니다.

관련 함수

변수	상호 작용
BrokenWeeks (page 206)	Qlik Sense 앱이 분리되지 않은 주와 함께 작동하는 경우 ReferenceDay 변수 설정이 적용됩니다. 단, 분리된 주를 사용하는 경우 1월 1일에 1주차가 시작되어 FirstWeekDay 변수 설정과 함께 종료되며 ReferenceDay 플래그는 무시됩니다.
FirstWeekDay (page 219)	주의 첫 번째 날로 사용할 날짜를 정의하는 정수입니다.

Qlik Sense는 ReferenceDay에 다음 값을 설정할 수 있습니다.

ReferenceDay 값

Value	기준일
0(기본값)	1월 4일
1	1월 1일
2	1월 2일
3	1월 3일
4	1월 4일
5	1월 5일
6	1월 6일
7	1월 7일

다음 예에서 ReferenceDay = 3은 1월 3일을 기준일로 정의합니다.

```
SET ReferenceDay=3; //(set January 3 as the reference day)
```

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

주 및 주차에 대해 ISO 설정을 사용하려는 경우 스크립트에 다음이 포함되어 있는지 확인합니다.

```
Set FirstWeekDay=0;
Set BrokenWeeks=0;
Set ReferenceDay=4;    // Jan 4th is always in week 1
```

미국 설정을 사용하려는 경우 스크립트에 다음이 포함되어 있는지 확인합니다.

```
Set FirstWeekDay=6;
Set BrokenWeeks=1;
Set ReferenceDay=1;     // Jan 1st is always in week 1
```

예 1 - 기본값 ReferenceDay=0을 사용한 로드 스크립트

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 0으로 설정된 ReferenceDay 변수.
- 앱이 분리되지 않은 주를 사용하도록 BrokenWeeks 변수를 0으로 설정합니다.
- 2019년 말부터 2020년 초까지의 날짜 데이터 집합.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- week
- weekday

결과 테이블

date	주	weekday
12/27/2019	52	Fri
12/28/2019	52	Sat
12/29/2019	1	Sun
12/30/2019	1	Mon
12/31/2019	1	Tue
01/01/2020	1	Wed
01/02/2020	1	Thu
01/03/2020	1	Fri
01/04/2020	1	Sat
01/05/2020	2	Sun
01/06/2020	2	Mon
01/07/2020	2	Tue
01/08/2020	2	Wed

date	주	weekday
01/09/2020	2	Thu
01/10/2020	2	Fri
01/11/2020	2	Sat

52주차는 12월 28일 토요일에 끝납니다. ReferenceDay에 따라 1월 4일이 1주차에 포함되어야 하므로 1주차는 12월 29일에 시작하여 1월 4일 토요일에 끝납니다.

예 - ReferenceDay 변수가 5로 설정됨

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 5으로 설정된 ReferenceDay 변수.
- 앱이 분리되지 않은 주를 사용하도록 BrokenWeeks 변수를 0으로 설정합니다.
- 2019년 말부터 2020년 초까지의 날짜 데이터 집합.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- week
- weekday

결과 테이블

date	주	weekday
12/27/2019	52	Fri
12/28/2019	52	Sat
12/29/2019	53	Sun
12/30/2019	53	Mon
12/31/2019	53	Tue
01/01/2020	53	Wed
01/02/2020	53	Thu
01/03/2020	53	Fri
01/04/2020	53	Sat
01/05/2020	1	Sun
01/06/2020	1	Mon
01/07/2020	1	Tue
01/08/2020	1	Wed
01/09/2020	1	Thu
01/10/2020	1	Fri
01/11/2020	1	Sat

52주차는 12월 28일 토요일에 끝납니다. BrokenWeeks 변수는 앱이 분리되지 않은 주를 사용하도록 합니다. 기준일 값을 5로 사용하려면 1월 5일이 1주차에 포함되어야 합니다.

그러나 이는 전년도 52주차 종료 8일 후입니다. 따라서 53주차는 12월 29일에 시작하여 1월 4일에 끝납니다. 1주차는 1월 5일 일요일에 시작됩니다.

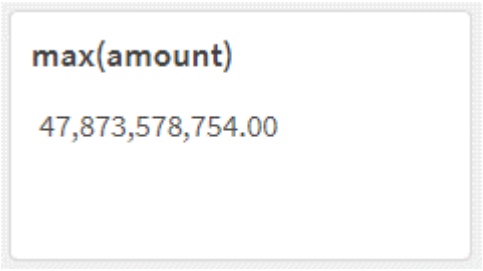
ThousandSep

정의된 천 단위 구분 기호에 따라 운영 체제의 자릿수 구분 기호를 바꿉니다(국가별 설정).

구문:

```
ThousandSep
```

ThousandSep 변수를 사용하는 Qlik Sense 개체(천 단위 구분 기호 포함)



Qlik Sense 앱은 이 서식 지정을 따르는 텍스트 필드를 숫자로 해석합니다. 이 서식 지정은 숫자 필드의 **숫자 서식** 지정 속성이 **숫자**로 설정된 경우 차트 개체에 표시됩니다.

ThousandSep는 여러 지역 설정에서 받은 데이터 소스를 처리할 때 유용합니다.

> *응용 프로그램에서 개체가 이미 만들어지고 서식이 지정된 후에 ThousandSep 변수가 수정되면 사용자는 **숫자 서식 지정** 속성 **숫자**를 선택 취소한 다음 다시 선택하여 각 관련 필드의 서식을 다시 지정해야 합니다.*

다음 예에서는 ThousandSep 시스템 변수의 사용 사례를 보여 줍니다.

```
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)
```

다음 항목은 이 함수를 사용하는 데 도움이 될 수 있습니다.

관련 항목

항목	설명
DecimalSep (page 217)	텍스트 필드 해석의 경우 이 함수에서 제공하는 소수점 구분 기호 설정도 적용되어야 합니다. 숫자 서식 지정의 경우 **DecimalSep**는 필요에 따라 Qlik Sense에서 사용됩니다.

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

예 1 - 기본 시스템 변수

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Transactions라는 테이블에 로드되는 데이터 집합.
- 기본 ThousandSep 변수 정의를 사용합니다.

로드 스크립트

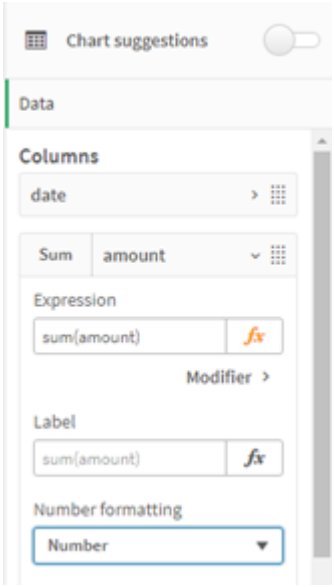
```
Transactions:
Load
date,
id,
amount
Inline
[
date,id,amount
01/01/2022,1,10000000441
01/02/2022,2,21237492432
01/03/2022,3,41249475336
01/04/2022,4,24313369837
01/05/2022,5,47873578754
01/06/2022,6,24313884663
01/07/2022,7,28545883436
01/08/2022,8,35545828255
01/09/2022,9,37565817436
01/10/2022,10,3454343566
];
```

결과

다음과 같이 하십시오.

1. 데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다.date.
2. 다음 측정값을 추가합니다.
   `=sum(amount)`
3. 속성 패널의 **데이터**에서 측정값을 선택합니다.
4. **숫자 서식 지정**에서 **숫자**를 선택합니다.

차트 측정값의 숫자 서식 지정 조정



결과 테이블

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

이 예에서는 쉼표 서식(',')으로 설정된 기본 ThousandSep 정의가 사용됩니다. 결과 테이블에서 금액 필드의 서식을 사용하면 천 단위 그룹 사이에 쉼표를 표시합니다.

예 2 - 시스템 변수 변경

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Transactions라는 테이블에 로드된 첫 번째 예의 동일한 데이터 집합.
- 천 단위 구분 기호로 '*' 문자를 표시하도록 스크립트 시작 시 ThousandSep 정의 수정. 이는 극단적인 예이며 변수의 기능을 보여 주기 위해서만 사용됩니다.

이 예에서 사용된 수정은 극단적이며 일반적으로 사용되지 않지만 변수의 기능을 보여 주기 위해 여기에 표시됩니다.

로드 스크립트

```
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
];
```

결과

다음과 같이 하십시오.

1. 데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다.date.
2. 다음 측정값을 추가합니다.
   `=sum(amount)`
3. 속성 패널의 **데이터**에서 측정값을 선택합니다.
4. **숫자 서식 지정**에서 **사용자 지정**을 선택합니다.

결과 테이블

date	=sum(amount)
01/01/2022	10*000*000*441.00
01/02/2022	21*237*492*432.00

date	=sum(amount)
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

스크립트 시작 시 ThousandSep 시스템 변수가 '*'로 수정됩니다. 결과 테이블에서 금액 필드의 서식을 사용하면 천 단위 그룹 사이에 '*'를 표시하는 것을 볼 수 있습니다.

예 3 - 텍스트 해석

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Transactions라는 테이블에 로드되는 데이터 집합.
- 쉼표가 천 단위 구분 기호로 사용되는 텍스트 서식의 숫자 필드가 있는 데이터.
- 기본 ThousandSep 시스템 변수를 사용합니다.

로드 스크립트

```
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'
];
```

결과

다음과 같이 하십시오.

1. 데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다.date.
2. 다음 측정값을 추가합니다.
   `=sum(amount)`
3. 속성 패널의 **데이터**에서 측정값을 선택합니다.
4. **숫자 서식 지정**에서 **숫자**를 선택합니다.
5. 금액 필드가 숫자 값인지 여부를 평가하려면 다음 측정값을 추가합니다.
   `=isnum(amount)`

결과 테이블

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

데이터가 로드되면 ThousandSep 변수에 맞는 데이터로 인해 Qlik Sense가 금액 필드를 숫자 값으로 해석한 것을 알 수 있습니다. 이는 각 항목을 -1 또는 TRUE로 평가하는 isnum() 함수에서 보여 줍니다.

Qlik Sense에서 부울 true 값은 -1로 표시되고 false 값은 0으로 표시됩니다.

TimeFormat

정의된 서식에 따라 운영 체제의 시간 서식을 바꿉니다(국가별 설정).

구문:

```
TimeFormat
```

```
Set TimeFormat='hh:mm:ss';
```

TimestampFormat

정의된 서식에 따라 운영 체제의 날짜 및 시간 서식을 바꿉니다(국가별 설정).

구문:

```
TimestampFormat
```

다음 예에서는 다른 **SET TimestampFormat** 문의 결과를 표시하기 위해 *1983-12-14T13:15:30Z*를 타임스탬프 데이터로 사용합니다. 사용된 날짜 서식은 **YYYYMMDD**이고 시간 서식은 **h:mm:ss TT**입니다. 데이터 로드 스크립트 맨 위에 날짜 서식은 **SET DateFormat** 문으로 지정되고 시간 서식은 **SET TimeFormat** 문으로 지정되어 있습니다.

결과

예	결과
`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

예: 로드 스크립트

예: 로드 스크립트

첫 번째 로드 스크립트에서는 *SET TimestampFormat='DD/MM/YYYY h:mm:ss[.fff] TT'*가 사용됩니다. 두 번째 로드 스크립트에서는 타임스탬프 서식이 *SET TimestampFormat='MM/DD/YYYY hh:mm:ss[.fff]'*로 변경되었습니다. 서로 다른 결과를 통해 **SET TimeFormat** 문이 다른 시간 데이터 서식으로 작동하는 방식을 알 수 있습니다.

아래 테이블은 이후 로드 스크립트에서 사용되는 데이터 집합을 보여 줍니다. 테이블의 두 번째 열은 데이터 집합의 각 타임스탬프 서식을 보여 줍니다. 처음 다섯 개의 타임스탬프는 ISO 8601 규칙을 따르지만 여섯 번째는 따르지 않습니다.

데이터 집합

사용된 시간 데이터와 데이터 집합의 각 타임스탬프에 대한 서식을 보여 주는 테이블입니다.

transaction_timestamp	time data format
2018-08-30	YYYY-MM-DD
20180830T193614.857	YYYYMMDDhhmmss.sss
20180830T193614.857+0200	YYYYMMDDhhmmss.sss±hhmm

transaction_timestamp	time data format
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

데이터 로드 편집기에서 새 섹션을 만든 다음 예제 스크립트를 추가하고 실행합니다. 그런 다음, 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.

로드 스크립트

```
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
];
```

결과

로드 스크립트에서 사용하는 TimestampFormat 해석 변수의 결과를 보여주는 Qlik Sense 테이블입니다. 데이터 집합의 마지막 타임스탬프가 올바른 날짜를 반환하지 않습니다.

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

transaction_id	transaction_timestamp	LogTimeStamp
3754	2018-09-16T13:15:30Z	2018-09-16 13:15:30
3755	9/30/18 19:36:14	-

다음 로드 스크립트는 동일한 데이터 집합을 사용합니다. 그러나 여섯 번째 타임스탬프의 ISO 8601이 아닌 서식과 일치시키는 데 *SET TimestampFormat='MM/DD/YYYY hh:mm:ss[.fff]'*를 사용합니다.

데이터 로드 편집기에서 이전 예제 스크립트를 아래 스크립트로 바꾸고 실행합니다. 그런 다음, 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.

로드 스크립트

```
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
];
```

결과

로드 스크립트에서 사용하는 TimestampFormat 해석 변수의 결과를 보여 주는 Qlik Sense 테이블입니다.

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

transaction_id	transaction_timestamp	LogTimeStamp
3754	2018-09-16T13:15:30Z	2018-09-16 13:15:30
3755	9/30/18 19:36:14	2018-09-16 19:36:14

4.9 Direct Discovery 변수

Direct Discovery 시스템 변수

DirectCacheSeconds
시각화에 대한 Direct Discovery 쿼리 결과에 캐싱 한계를 설정할 수 있습니다. 이 한계에 도달하면 새로운 Direct Discovery 쿼리가 수행될 때 Qlik Sense가 캐시를 지웁니다. Qlik Sense는 지정된 제한 시간 동안 소스 데이터에 선택 내용을 쿼리하고 다시 캐시를 만듭니다. 선택 내용의 각 조합 결과는 개별적으로 캐시됩니다. 즉, 각 선택 내용에 대해 개별적으로 캐시가 새로 고침되므로 한 번 선택하면 선택한 필드에만 해당하는 캐시가 새로 고침되고, 두 번째로 선택하면 관련 필드에 해당하는 캐시가 새로 고침됩니다. 두 번째 선택에 첫 번째 선택에서 새로 고침된 필드가 포함된 경우 캐싱 한계에 도달하지 않았다면 캐시에서 해당 필드가 다시 업데이트되지 않습니다.

Direct Discovery 캐시는 **테이블** 시각화에 적용되지 않습니다. 테이블 선택은 매번 데이터 소스를 쿼리합니다.

한계 값은 초 단위로 설정해야 합니다. 기본 캐시 한계는 1800초(30분)입니다.

DirectCacheSeconds에 사용되는 값은 **DIRECT QUERY** 문이 실행될 때 설정된 값입니다. 실행 중에 이 값을 변경할 수는 없습니다.

```
SET DirectCacheSeconds=1800;
```

DirectConnectionMax
연결 풀링 기능을 사용하여 데이터베이스에 대한 비동기, 병렬 호출을 수행할 수 있습니다. 풀링 기능을 설정하기 위한 로드 스크립트 구문은 다음과 같습니다.

```
SET DirectConnectionMax=10;
```
숫자 설정은 시트를 업데이트할 때 Direct Discovery 코드에서 사용해야 하는 데이터베이스 연결의 최대 수를 지정합니다. 기본 설정은 1입니다.

이 변수를 사용할 때 주의를 기울여야 합니다. 이를 1보다 크게 설정하면 Microsoft SQL Server에 연결할 때 문제를 일으키는 것으로 알려져 있습니다.

DirectUnicodeStrings
Direct Discovery는 특히 SQL Server와 같은 일부 데이터베이스에서 요구하는 대로 확장 문자 문자열 리터럴(N'<확장 문자열>')에 대한 SQL 표준 형식을 사용하여 확장 유니코드 데이터의 선택을 지원할 수 있습니다. 이 구문은 스크립트 변수 **DirectUnicodeStrings**를 통해 Direct Discovery에서 사용할 수 있습니다.

이 변수를 'true'로 설정하면 문자열 리터럴 앞에 ANSI 표준 와이드 문자 표식인 “N”을 사용할 수 있습니다. 데이터베이스에 따라 이 표준을 지원하지 않을 수 있습니다. 기본 설정은 'false'입니다.

DirectDistinctSupport
Qlik Sense 개체에서 **DIMENSION** 필드 값을 선택하면 소스 데이터베이스에 대한 쿼리가 생성됩니다. 쿼리에서 그룹화가 필요할 경우 Direct Discovery는 **DISTINCT** 키워드를 사용하여 고유한 값만 선택합니다. 하지만 일부 데이터베이스에는 **GROUP BY** 키워드가 필요합니다. 고유 값에 대한 쿼리에서 **DISTINCT** 대신 **GROUP BY**를 생성하려면 **DirectDistinctSupport**를 'false'로 설정하십시오.

```
SET DirectDistinctSupport='false';
```
DirectDistinctSupport가 true로 설정된 경우 **DISTINCT**가 사용됩니다. 설정하지 않았을 때의 기본 동작은 **DISTINCT**를 사용하는 것입니다.

DirectEnableSubquery
집합 크기가 큰 다중 테이블 시나리오의 경우, 대규모 IN 절을 만드는 대신 SQL 쿼리 내에 하위 쿼리를 만들 수 있습니다. 이 기능은 **DirectEnableSubquery**를 'true'로 설정하면 활성화됩니다. 기본값은 'false'입니다.

> ***DirectEnableSubquery**를 활성화하면 Direct Discovery 모드가 아닌 테이블을 로드할 수 없습니다.*

```
SET DirectEnableSubquery='true';
```

Teradata 쿼리 구간 설정 변수

Teradata 쿼리 구간 설정은 회계, 우선 순위 지정 및 워크로드 관리를 개선하기 위해 원본 Teradata 데이터베이스와 엔터프라이즈 응용 프로그램의 공동 작업을 지원하는 기능입니다. 쿼리 구간 설정을 사용하면 쿼리 주변에 사용자 자격 증명과 같은 메타데이터를 래핑할 수 있습니다.

두 가지 변수를 사용할 수 있는데, 둘 다 평가 후 데이터베이스로 전송되는 문자열입니다.

SQLSessionPrefix
이 문자열은 데이터베이스에 대한 연결이 생성되면 전송됩니다.

```
SET SQLSessionPrefix = 'SET QUERY_BAND = ' & Chr(39) & 'Who=' & OSuser() & ';' & Chr(39) & '
FOR SESSION;';
```
예를 들어 **OSuser()**가 *WA\sbt*를 반환할 경우 `SET QUERY_BAND = 'Who=WA\sbt;' FOR SESSION;`으로 평가되고, 데이터베이스 연결이 생성되면 데이터베이스로 전송됩니다.

SQLQueryPrefix
이 문자열은 각 단일 쿼리에 대해 전송됩니다.

```
SET SQLSessionPrefix = 'SET QUERY_BAND = ' & Chr(39) & 'Who=' & OSuser() & ';' & Chr(39) & '
FOR TRANSACTION;';
```

Direct Discovery 문자 변수

DirectFieldColumnDelimiter
쉼표 이외의 문자를 필드 구분 기호로 사용해야 하는 데이터베이스에 대한 **Direct Query** 문에서 필드 구분 기호로 사용할 문자를 설정할 수 있습니다. 지정된 문자는 **SET** 문에서 작은따옴표로 묶어야 합니다.

```
SET DirectFieldColumnDelimiter= '|'
```

DirectStringQuoteChar
생성된 쿼리에서 문자열을 인용하는 데 사용할 문자를 지정할 수 있습니다. 기본값은 작은따옴표입니다. 지정된 문자는 **SET** 문에서 작은따옴표로 묶어야 합니다.

```
SET DirectStringQuoteChar= '"';
```

DirectIdentifierQuoteStyle
생성된 쿼리에서 사용할 식별자의 ANSI 이외 인용을 지정할 수 있습니다. 이때 사용 가능한 유일한 ANSI 이외 인용은 GoogleBQ입니다. 기본값은 ANSI입니다. 대문자, 소문자, 대/소문자 혼합을 사용할 수 있습니다 (ANSI, ansi, Ansi).

```
SET DirectIdentifierQuoteStyle="GoogleBQ";
```
예를 들어 다음 **SELECT** 문에 ANSI 인용이 사용되었습니다.

```
SELECT [Quarter] FROM [qvTest].[sales] GROUP BY [Quarter]
```
DirectIdentifierQuoteStyle을 "GoogleBQ"로 설정하면 **SELECT** 문에서 인용을 다음과 같이 사용합니다.

```
SELECT [Quarter] FROM [qvTest.sales] GROUP BY [Quarter]
```

DirectIdentifierQuoteChar
생성된 쿼리에서 식별자의 인용을 제어하는 문자를 지정할 수 있습니다. 이는 하나의 문자(예: 큰따옴표) 또는 2개의 문자(예: 대괄호 쌍)로 설정할 수 있습니다. 기본값은 큰따옴표입니다.

```
SET DirectIdentifierQuoteChar='[]';
SET DirectIdentifierQuoteChar='``';
SET DirectIdentifierQuoteChar=' ';
SET DirectIdentifierQuoteChar='""';
```

DirectTableBoxListThreshold
Direct Discovery 필드가 **테이블** 시각화에 사용되는 경우 표시되는 행의 수를 제한하기 위해 임계값이 설정됩니다. 기본 임계값은 1000 레코드입니다. 기본 임계값 설정은 로드 스크립트의 **DirectTableBoxListThreshold** 변수를 설정하여 변경할 수 있습니다. 예:

```
SET DirectTableBoxListThreshold=5000;
```
임계값 설정은 Direct Discovery 필드를 포함하는 **테이블** 시각화에만 적용됩니다. 인 메모리 필드만 포함하는 **테이블** 시각화에는 **DirectTableBoxListThreshold** 설정이 적용되지 않습니다.

임계값 제한보다 적은 수의 레코드가 선택될 때까지는 **테이블** 시각화에 필드가 표시되지 않습니다.

Direct Discovery 숫자 해석 변수

DirectMoneyDecimalSep
소수점 구분 기호로 정의된 변수로 Direct Discovery를 사용하여 데이터를 로드하도록 생성된 SQL 문의 통화에 대한 소수점 기호를 바꿉니다. 이 문자는 **DirectMoneyFormat**에서 사용한 문자와 일치해야 합니다.

기본값은 '.'입니다.

```
Set DirectMoneyDecimalSep='.';
```

DirectMoneyFormat
기호로 정의된 변수로 Direct Discovery를 사용하여 데이터를 로드하도록 생성된 SQL 문의 통화 서식을 바꿉니다. 천 단위 구분 기호에 대한 통화 기호는 포함할 수 없습니다.

기본값은 '#.0000'입니다.

```
Set DirectMoneyFormat='#.0000';
```

DirectTimeFormat
시간 서식으로 정의된 변수로 Direct Discovery을(를) 사용하여 데이터를 로드하도록 생성된 SQL 문의 시간 서식을 바꿉니다.

```
Set DirectTimeFormat='hh:mm:ss';
```

DirectDateFormat
날짜 서식으로 정의된 변수로 Direct Discovery를 사용하여 데이터를 로드하도록 생성된 SQL 문의 날짜 서식을 바꿉니다.

```
Set DirectDateFormat='MM/DD/YYYY';
```

DirectTimeStampFormat
서식으로 정의된 변수로 Direct Discovery를 사용하여 데이터를 로드하도록 생성된 SQL 문의 날짜 및 시간 서식을 바꿉니다.

```
Set DirectTimestampFormat='M/D/YY hh:mm:ss[.fff]';
```

4.10 오류 변수

스크립트 실행 후에는 모든 오류 변수에 값이 존재하게 됩니다. 첫 번째 변수인 ErrorMode는 사용자가 입력하는 값이고 마지막 3개는 Qlik Sense에서 출력되는 값으로, 스크립트 오류에 대한 정보가 포함되어 있습니다.

오류 변수 개요

각 변수는 개요 후에 자세히 설명합니다. 구문에서 변수 이름을 클릭하여 해당 특정 변수에 대한 세부 정보에 즉시 액세스할 수도 있습니다.

변수에 대한 자세한 내용은 Qlik Sense 온라인 도움말을 참조하십시오.

ErrorMode
이 오류 변수는 스크립트 실행 도중 오류가 발견되었을 때 Qlik Sense에서 실행할 동작을 결정합니다.

```
ErrorMode
```

ScriptError
이 오류 변수는 마지막으로 실행된 스크립트 문의 오류 코드를 반환합니다.

```
ScriptError
```

ScriptErrorCount
이 오류 변수는 현재 스크립트 실행 도중 오류를 일으킨 문의 총 수를 반환합니다. 이 변수는 스크립트 실행이 시작될 때 항상 0으로 초기화됩니다.

```
ScriptErrorCount
```

ScriptErrorList
이 오류 변수에는 마지막 스크립트 실행 도중에 발생한 모든 스크립트 오류의 연결 목록이 포함됩니다. 각각의 오류는 줄 바꿈으로 구분됩니다.

```
ScriptErrorList
```

ErrorMode

이 오류 변수는 스크립트 실행 도중 오류가 발견되었을 때 Qlik Sense에서 실행할 동작을 결정합니다.

구문:

```
ErrorMode
```

인수:

인수

인수	설명
ErrorMode=1	기본 설정입니다. 스크립트 실행이 정지되며 사용자에게 동작을 묻는 메시지가 표시됩니다(배치 모드가 아닐 경우).
ErrorMode =0	Qlik Sense에서 오류를 그냥 무시하고 다음 스크립트 문의 스크립트를 실행을 계속합니다.
ErrorMode =2	Qlik Sense에서 사용자에게 사전에 동작을 묻는 메시지를 표시하지 않고 "스크립트 실행에 실패했습니다..."라는 오류 메시지를 트리거합니다.

```
set ErrorMode=0;
```

ScriptError

이 오류 변수는 마지막으로 실행된 스크립트 문의 오류 코드를 반환합니다.

구문:

```
ScriptError
```

이 변수는 매번 스크립트 문이 성공적으로 실행된 후에 0으로 초기화됩니다. 오류가 발생하면 내부 Qlik Sense 오류 코드로 설정됩니다. 오류 코드는 숫자와 텍스트로 구성된 이중 값입니다. 다음과 같은 오류 코드가 있습니다.

스크립트 오류 코드

오류 코드	설명
0	오류가 없습니다. 이중 값 텍스트가 비어 있습니다.
1	일반 오류입니다.
2	구문 오류입니다.
3	일반 ODBC 오류입니다.
4	일반 OLE DB 오류입니다.
5	일반 사용자 지정 데이터베이스 오류입니다.
6	일반 XML 오류입니다.
7	일반 HTML 오류입니다.
8	파일이 없습니다.
9	데이터베이스가 없습니다.
10	테이블을 찾을 수 없습니다.
11	필드가 없습니다.
12	잘못된 파일 형식입니다.
16	의미 체계 오류입니다.

```
set ErrorMode=0;

LOAD * from abc.qvf;

if ScriptError=8 then

exit script;

//no file;

end if
```

ScriptErrorCount

이 오류 변수는 현재 스크립트 실행 도중 오류를 일으킨 문의 총 수를 반환합니다. 이 변수는 스크립트 실행이 시작될 때 항상 0으로 초기화됩니다.

구문:

```
ScriptErrorCount
```

ScriptErrorList

이 오류 변수에는 마지막 스크립트 실행 도중에 발생한 모든 스크립트 오류의 연결 목록이 포함됩니다. 각각의 오류는 줄 바꿈으로 구분됩니다.

구문:

```
ScriptErrorList
```

5 스크립트 표현식

표현식은 **LOAD** 문과 **SELECT** 문에 모두 사용될 수 있습니다. 여기에서 설명하는 구문과 함수는 **LOAD** 문에 적용되며, **SELECT** 문의 경우 Qlik Sense가 아니라 ODBC 드라이버에 의해 해석되므로 적용되지 않습니다. 하지만 대부분의 ODBC 드라이버는 종종 아래에 설명된 여러 함수를 해석할 수 있습니다.

표현식은 하나의 구문에 결합된 함수, 필드 및 연산자로 구성됩니다.

Qlik Sense 스크립트의 모든 표현식은 숫자 및/또는 문자열 중 적절한 항목을 반환합니다. 논리 함수 및 연산자는False에 대해 0, True에 대해 -1을 반환합니다. 숫자를 문자열로 변환하는 것과 그 반대의 변환은 암시적입니다. 논리 연산자와 함수는 0을 False로 해석하며 그 외 모든 것을 True로 해석합니다.

표현식의 일반적인 구문은 다음과 같습니다.

일반 구문

표현식	필드	연산자
expression ::= (constant	constant	|
expression ::= (constant	fieldref	|
expression ::= (constant	operator1 expression	|
expression ::= (constant	expression operator2 expression	|
expression ::= (constant	function	|
expression ::= (constant	(expression)	)

설명:

- **constant**는 곧은 작은따옴표로 묶인 문자열(텍스트, 날짜 또는 시간)이나 숫자입니다. 상수에는 천 단위 구분 기호가 사용되지 않으며, 소수점 구분 기호인 소수점이 사용됩니다.
- **fieldref**는 로드된 테이블의 필드 이름입니다.
- **operator1**은 단항 연산자입니다(오른쪽에 있는 하나의 표현식에서 작동).
- **operator2**는 이항 연산자입니다(양쪽에 하나씩 있는 두 개의 표현식에서 작동).
- **function ::= functionname(parameters)**
- **parameters ::= expression { , expression }**

파라메타의 수와 유형은 임의가 아니라, 사용되는 함수에 따라 결정됩니다.

따라서 표현식과 함수는 자유롭게 중첩될 수 있으며, 표현식이 해석 가능한 값을 반환하는 한 Qlik Sense는 오류 메시지를 표시하지 않습니다.

6 차트 표현식

차트(시각화) 표현식은 함수, 필드, 수학 연산자 (+ * / =) 및 기타 측정값의 조합입니다. Qlik Sense 명령줄 구문과 스크립트 구문에는 Backus-Naur 이론(또는 BNF 코드)이라는 표기법이 사용됩니다. 표현식은 시각화에 표시할 수 있는 결과를 생산하기 위해 앱에 있는 데이터를 처리하는 데 사용됩니다. 표현식은 측정값에서 사용하는 것으로 제한되지 않습니다. 제목, 부제, 각주와 더불어 차원에 대한 표현식을 사용하여 더 동적이고 강력한 시각화를 만들 수 있습니다.

예를 들어, 이는 시각화의 제목을 일정한 텍스트로 정하는 것이 아니라, 선택 내용에 따라 결과가 변경되는 표현식으로 만들 수 있음을 의미합니다.

스크립트 함수 및 차트 함수에 대한 더 자세한 내용은 스크립트 구문 및 차트 함수을 참조하십시오.

6.1 집계 범위 정의

표현식에서 집계의 값을 정의하는 데 사용되는 레코드는 일반적으로 두 가지 요소가 함께 결정합니다. 시각화에서 사용되는 이 두 요소는 다음과 같습니다.

- (차트 표현식 내 집계의) 차원 값
- 선택

이 요소는 함께 집계의 범위를 정의합니다. 계산에서 선택, 차원 또는 두 가지 모두 무시하는 것이 바람직한 상황이 있을 수 있습니다. 차트 함수에서 TOTAL 한정자, 집합 분석 또는 이 둘의 조합을 사용하여 이를 구현할 수 있습니다.

집계: 방법 및 설명

방법	설명
TOTAL 한정자	집계 함수 내에서 total 한정자를 사용하여 차원 값을 무시할 수 있습니다. 가능한 모든 필드 값에 대해 집계가 수행됩니다. **TOTAL** 한정자 뒤에는 꺾쇠 괄호로 묶인 하나 이상의 필드 이름 목록이 올 수 있습니다. 이러한 필드 이름은 차트 차원 변수의 하위 집합이어야 합니다. 이 경우 나열된 차원을 제외하고 모든 차트 차원 변수를 무시하면서 계산이 실행됩니다. 즉, 나열된 차원 필드 내 필드 값의 각 조합에 대해 하나의 값이 반환됩니다. 또한 현재 차트 내의 차원이 아닌 필드가 목록에 포함될 수 있습니다. 이는 차원 필드가 고정되지 않은 그룹 차원의 경우에 유용할 수 있습니다. 그룹 내의 모든 변수를 나열할 경우 드릴다운 수준이 변화할 때 함수가 작동하게 됩니다.
집합 분석	집계 내에서 집합 분석을 사용하여 선택 내용을 재정의할 수 있습니다. 여러 차원에 분산된 모든 값에 대해 집계가 수행됩니다.

방법	설명
TOTAL 한정자 및 집합 분석	집계 내에서 **TOTAL** 한정자 및 집합 분석을 사용하여 선택 내용을 재정의하고 차원을 무시할 수 있습니다.

ALL 한정자	집계 내에서 **ALL** 한정자를 사용하여 선택 내용과 차원을 무시할 수 있습니다. {1} 집합 분석 문과 **TOTAL** 한정자를 사용하면 동일한 기능을 구현할 수 있습니다.
	`=sum(All Sales)`
	`=sum({1} Total Sales)`

TOTAL 한정자

다음 예는 TOTAL 한정자를 사용하여 상대적 비율을 계산하는 방법을 보여 줍니다. Q2가 선택되었다고 가정할 때, TOTAL을 사용하여 차원을 무시하고 모든 값의 합계를 계산합니다.

예: TOTAL 한정자

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%

숫자를 백분율로 표시하려면 속성 패널에서 백분율 값으로 표시할 측정값에 대해 ***Number formatting****에서* ***Number****를 선택하고* ***Formatting****에서 % 서식 중 하나와* ***Simple****을 선택합니다.*

집합 분석

다음 예는 집합 분석을 사용하여 선택을 실행하기 전에 데이터 셋을 비교하는 방법을 보여줍니다. Q2가 선택되었다고 가정할 때, 집합 정의 {1}의 집합 분석을 사용하여 모든 선택을 무시하고 차원으로 분할된 모든 값의 합계를 계산합니다.

예: 집합 분석

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%

Year	Quarter	Sum(Amount)	Sum({1} Amount)	Sum(Amount)/Sum({1} Amount)
2013	Q3	0	1100	0%
2013	Q4	0	1400	0%
2013	Q2	1300	1300	100%

TOTAL 한정자 및 집합 분석

다음 예는 집합 분석과 TOTAL 한정자를 함께 사용하여 선택을 실행하기 전에 모든 차원에 걸쳐 데이터 셋을 비교하는 방법을 보여 줍니다. Q2가 선택되었다고 가정할 때, 집합 정의 {1}의 집합 분석과 TOTAL 한정자를 사용하여 모든 선택 내용과 차원을 무시하고 모든 값의 합계를 계산합니다.

예: TOTAL 한정자 및 집합 분석

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%

예에서 사용된 데이터:

```
AggregationScope:
LOAD * inline [
Year Quarter Amount
2012 Q1 1100
2012 Q2 1700
2012 Q3 1400
2012 Q4 1800
2013 Q1 1000
2013 Q2 1300
2013 Q3 1100
2013 Q4 1400] (delimiter is ' ');
```

6.2 집합 분석

앱에서 선택하면 데이터에 있는 레코드의 하위 집합을 정의합니다. Sum(), Max(), Min(), Avg() 및 Count()와 같은 집계 함수는 이 하위 집합을 기반으로 계산됩니다.

즉, 선택 내용에 따라 집계 범위가 정의됩니다. 계산이 수행되는 레코드 집합을 정의합니다.

집합 분석은 현재 선택에 의해 정의된 레코드 집합과 다른 범위를 정의하는 방법을 제공합니다. 이 새로운 범위는 대체 선택으로 간주될 수도 있습니다.

이는 현재 선택을 특정 값(예: 작년 값 또는 글로벌 시장 점유율)과 비교하려는 경우에 유용할 수 있습니다.

집합 표현식

집합 표현식은 집계 함수 내부 및 외부에서 사용할 수 있으며 중괄호로 묶입니다.

내부 집합 표현식

```
Sum( {$<Year={2021}>} Sales )
```

외부 집합 표현식

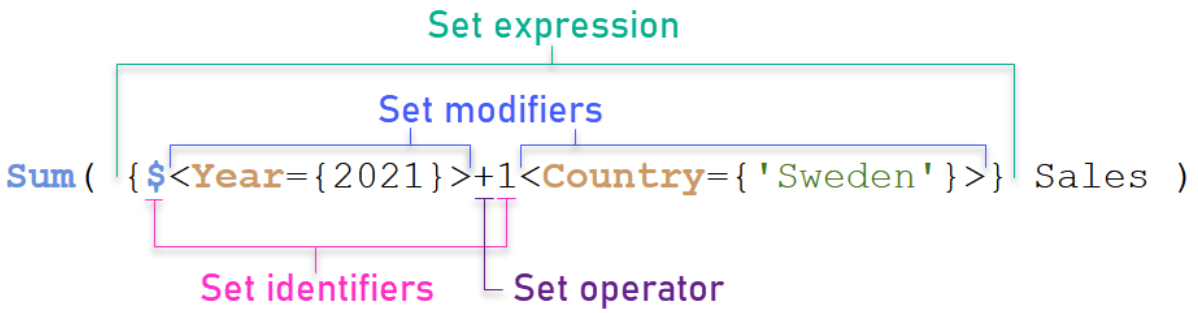
```
{<Year={2021}>} Sum(Sales) / Count(distinct Customer)
```

집합 표현식은 다음 요소의 조합으로 구성됩니다.

- **식별자** 집합 식별자는 다른 곳에서 정의된 선택을 나타냅니다. 또한 데이터의 특정 레코드 집합을 나타냅니다. 현재 선택, 북마크에서 선택 또는 대체 상태에서 선택일 수 있습니다. 단순한 집합 표현식은 현재 선택 내용에 포함된 모든 레코드를 의미하는 달러 기호, {$} 등과 같은 하나의 식별자로 구성됩니다.
  예: $, 1, BookMark1, State2
- **연산자** 집합 연산자를 사용하여 서로 다른 집합 식별자 간의 합집합, 차집합 또는 교집합을 만들 수 있습니다. 이렇게 하면 집합 식별자로 정의된 선택 내용의 하위 집합 또는 상위 집합을 만들 수 있습니다.
  예: +, -, *, /
- **수정자** 집합 식별자에 집합 수정자를 추가하여 선택을 변경할 수 있습니다. 수정자는 자체적으로 사용할 수도 있으며 기본 식별자를 수정합니다. 수정자는 꺾쇠 괄호 <…>로 묶어야 합니다.
  예: <Year={2020}>, <Supplier={ACME}>

요소가 결합되어 집합 표현식을 형성합니다.

집합 표현식의 요소



예를 들어 위의 집합 표현식은 집계 Sum(Sales)에서 작성됩니다.

첫 번째 피연산자는 현재 선택에 대한 2021년의 판매량을 반환합니다. 이는 $ 집합 식별자와 2021년의 선택을 포함하는 수정자로 표시됩니다. 두 번째 피연산자는 Sweden에 대해 Sales를 반환하고 1 집합 식별자로 표시되는 현재 선택을 무시합니다.

마지막으로 표현식은 + 집합 연산자로 표시된 대로 두 집합 피연산자 중 하나에 속하는 레코드로 구성된 집합을 반환합니다.

예

위의 집합 표현식 요소를 결합하는 예는 다음 항목에서 사용할 수 있습니다.

자연적 집합

일반적으로 집합 표현식은 데이터 모델의 레코드 집합과 이 데이터 하위 집합을 정의하는 선택 내용을 모두 나타냅니다. 이 경우 집합을 자연적 집합이라고 합니다.

집합 수정자가 있든 없든 집합 식별자는 항상 자연적 집합을 나타냅니다.

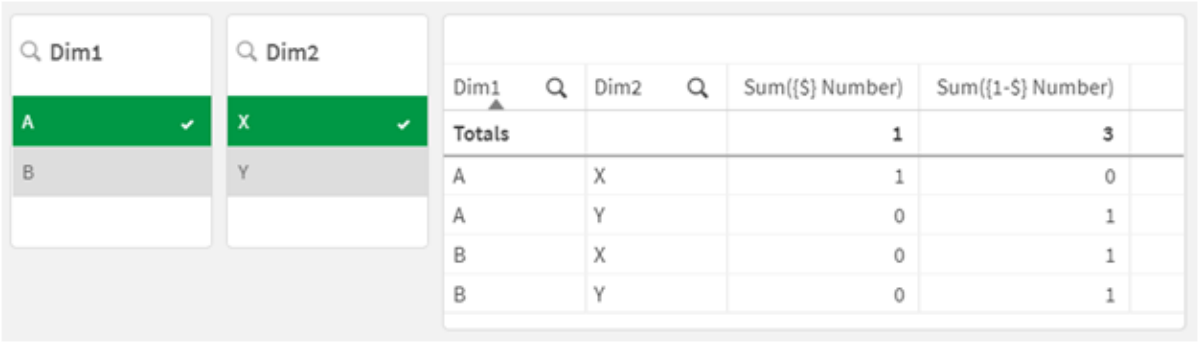
그러나 집합 연산자를 사용하는 집합 표현식은 레코드의 하위 집합을 나타내기도 하지만 일반적으로 필드 값 선택을 사용하여 설명할 수는 없습니다. 이러한 표현은 비자연적 집합입니다.

예를 들어 {1-$}에 의해 제공된 집합이 항상 선택 내용으로 정의될 수는 없습니다. 따라서 자연적 집합이 아닙니다. 이는 다음 데이터를 로드하고 테이블에 추가한 다음 필터 창을 사용하여 선택하여 표시할 수 있습니다.

```
Load * Inline
[Dim1, Dim2, Number
A, X, 1
A, Y, 1
B, X, 1
B, Y, 1];
```

Dim1 및 Dim2에 대해 선택하면 다음 표에 표시된 보기가 표시됩니다.

자연적 집합 및 비자연적 집합이 있는 테이블



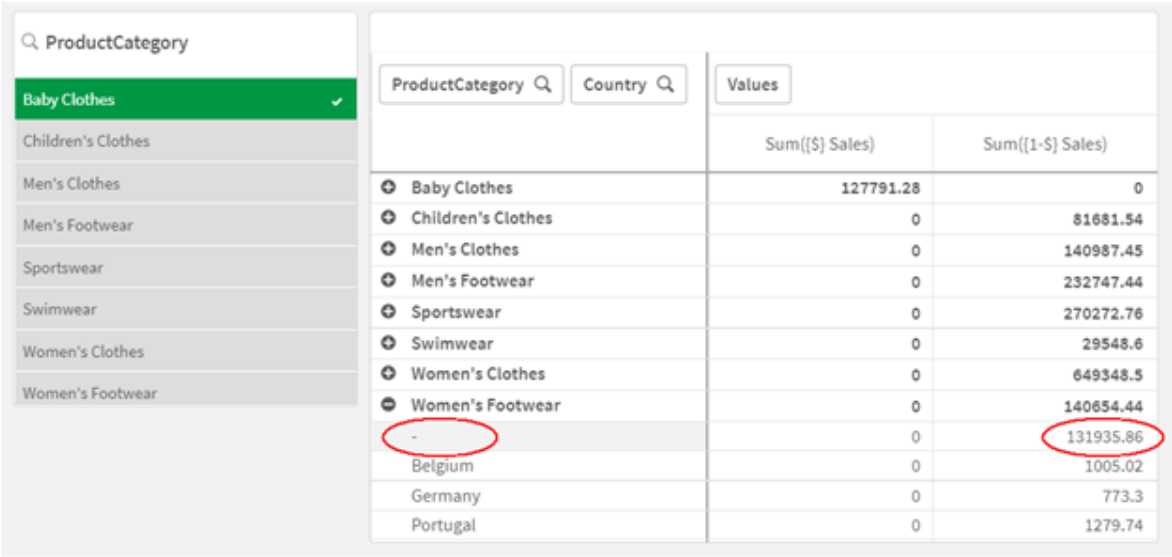
첫 번째 측정값의 집합 표현식은 {$}로 선택한 내용에 해당하는 자연적 집합을 사용합니다.

두 번째 측정값은 다릅니다. {1-$}를 사용합니다. 이 집합에 해당하는 선택은 불가능하므로 비자연적 집합입니다.

이러한 구분은 다음과 같은 여러 가지 결과가 생성됩니다.

- 집합 수정자는 집합 식별자에만 적용할 수 있습니다. 임의의 집합 표현식에는 적용할 수 없습니다. 예를 들어 다음과 같은 집합 표현식을 사용할 수 없습니다.
  `{ (BM01 * BM02) <Field={x,y}> }`
  여기에서 일반(둥근) 괄호는 집합 수정자가 적용되기 전에 BM01과 BM02 사이의 교차 집합을 평가해야 함을 의미합니다. 그 이유는 수정할 수 있는 요소 집합이 없기 때문입니다.
- P() 및 E() 요소 함수 내에서 비자연적 집합을 사용할 수 없습니다. 이 함수는 요소 집합을 반환하지만 비자연적 집합에서 요소 집합을 추론하는 것은 불가능합니다.

- 데이터 모델에 많은 테이블이 있는 경우 비자연적 집합을 사용하는 측정값이 항상 올바른 차원 값에 기여하지는 않습니다. 예를 들어, 다음 차트에서 일부 제외된 판매 수가 올바른 Country로 기여하는 반면 다른 경우에는 NULL이 Country로 포함됩니다.
  비자연적 집합이 있는 차트



  할당이 올바르게 수행되었는지 여부는 데이터 모델에 따라 다릅니다. 이 경우 선택에서 제외된 국가에 해당하는 번호는 할당할 수 없습니다.

식별자	설명
1	선택 내용과 관계없이 응용 프로그램에 있는 모든 레코드의 전체 집합을 나타냅니다.
$	현재 선택의 레코드를 나타냅니다. 따라서 집합 표현식 **{$}**는 집합 표현식을 지정하지 않는 것과 동일합니다.
$1	이전 선택 내용을 나타냅니다. $2는 하나를 제외한 이전 선택 내용을 나타냅니다.
$_1	다음(이후) 선택 내용을 나타냅니다. $_2는 하나를 제외한 다음 선택 내용을 나타냅니다.
BM01	북마크 ID 또는 북마크 이름을 사용할 수 있습니다.
MyAltState	상태 이름을 사용하면 대체 상태에서 선택한 내용을 참조할 수 있습니다.

예	결과
sum ({1} Sales)	선택 내용은 무시하되 차원은 무시하지 않고 앱에 해당하는 총 판매량을 반환합니다.
sum ({$} Sales)	현재 선택의 판매량을 반환하며, sum(Sales)과 동일합니다.
sum ({$1} Sales)	이전 선택에 해당하는 판매량을 반환합니다.
sum ({BM01} Sales)	*BM01*이란 이름의 북마크에 해당하는 판매량을 반환합니다.

예	결과
sum({$<OrderDate = DeliveryDate>} Sales)	OrderDate = DeliveryDate일 때 현재 선택의 판매량을 반환합니다.
sum({1<Region = {US}>} Sales)	현재 선택 내용을 무시하고 미국 지역의 판매량을 반환합니다.
sum({$<Region = >} Sales)	*Region*의 선택을 제거하고 선택의 판매량을 반환합니다.
sum({<Region = >} Sales)	위 예와 동일한 내용을 반환합니다. 수정할 집합이 생략된 경우 $로 간주됩니다.
sum({$<Year={2000}, Region={“U*”}>} Sales)	*Year* 및 *Region* 모두에 새로운 선택 내용을 적용하여 현재 선택의 판매량을 반환합니다.

집합 식별자

집합 식별자는 데이터의 레코드 집합(모든 데이터 또는 데이터의 하위 집합)을 나타냅니다. 선택으로 정의된 레코드 집합입니다. 현재 선택, 모든 데이터(선택 없음), 북마크에서 선택 또는 대체 상태에서 선택일 수 있습니다.

예 `Sum( {$<Year = {2009}>} Sales )`에서 식별자는 달러 기호 $입니다. 이는 현재 선택을 나타냅니다. 또한 가능한 모든 레코드를 나타냅니다. 그런 다음 이 집합은 집합 표현식의 수정자 부분에 의해 변경될 수 있습니다. `Year`에서 선택 2009가 추가됩니다.

더 복잡한 집합 표현식에서는 두 개의 식별자를 연산자와 함께 사용하여 두 레코드 집합의 합집합, 차집합 또는 교집합을 형성할 수 있습니다.

다음 표는 몇 가지 사용되는 식별자를 보여 줍니다.

일반 식별자가 있는 예

식별자	설명
1	선택 내용과 관계없이 응용 프로그램에 있는 모든 레코드의 전체 집합을 나타냅니다.
$	기본 상태에서 현재 선택의 레코드를 나타냅니다. 따라서 {$} 집합 표현식은 일반적으로 집합 표현식을 지정하지 않는 것과 같습니다.
$1	기본 상태의 이전 선택을 나타냅니다. $2는 하나를 제외한 이전 선택을 나타냅니다.
$_1	다음(이후) 선택을 나타냅니다. $_2는 하나를 제외한 다음 선택을 나타냅니다.
BM01	북마크 ID 또는 북마크 이름을 사용할 수 있습니다.
AltState	상태 이름으로 대체 상태를 참조할 수 있습니다.
AltState::BM01	북마크에는 모든 상태의 선택이 포함되며 북마크 이름을 한정하여 특정 북마크를 참조할 수 있습니다.

다음 표는 다른 식별자가 포함된 예를 보여 줍니다.

다양한 식별자를 사용하는 예

예	결과
Sum ({1} Sales)	선택 내용은 무시하되 차원은 무시하지 않고 앱에 해당하는 총 판매량을 반환합니다.
Sum ({$} Sales)	현재 선택의 판매량을 반환하며, Sum(Sales)과 동일합니다.
Sum ({$1} Sales)	이전 선택에 해당하는 판매량을 반환합니다.
Sum ({BM01} Sales)	BM01이란 이름의 북마크에 해당하는 판매량을 반환합니다.

집합 연산자

집합 연산자는 데이터 집합을 포함, 제외 또는 교차하는 데 사용됩니다. 모든 연산자는 집합을 피연산자로 사용하며 집합을 결과로 반환합니다.

두 가지 다른 상황에서 집합 연산자를 사용할 수 있습니다.

- 데이터의 레코드 집합을 나타내는 집합 식별자에 대한 집합 작업을 수행합니다.
- 요소 집합, 필드 값 또는 집합 수정자 내부에서 집합 작업을 수행합니다.

다음 표는 집합 표현식에서 사용할 수 있는 연산자를 보여 줍니다.

연산자

연산자	설명
+	합집합. 이 2항 연산은 두 집합 피연산자 중 하나에라도 속하는 레코드 또는 요소로 구성된 집합을 반환합니다.
-	차집합. 이 2항 연산은 두 집합 피연산자 중 첫 번째 피연산자에 속하지만 다른 피연산자에는 속하지 않는 레코드 또는 요소로 구성된 집합을 반환합니다. 또한 단항 연산자로 사용된 경우 여집합을 반환합니다.
*	교집합. 이 2항 연산은 두 집합 피연산자 모두에 속하는 레코드 또는 요소로 구성된 집합을 반환합니다.
/	대칭차(XOR). 이 2항 연산은 두 집합 피연산자 중 하나에 속하지만 두 집합 피연산자 모두에 속하지 않는 레코드 또는 요소로 구성된 집합을 반환합니다.

다음 표는 연산자가 있는 예를 보여 줍니다.

연산자를 사용한 예

예	결과
Sum ({1-$} Sales)	현재 선택에서 제외된 모든 항목에 대한 판매량을 반환합니다.
Sum ({$*BM01} Sales)	선택 및 북마크 #160;BM01 사이의 교집합에 대한 판매량을 반환합니다.
Sum ({-($+BM01)} Sales)	선택 및 BM01 북마크에 의해 제외된 판매량을 반환합니다.

예	결과
`Sum ({$<Year={2009}>+1<Country={'Sweden'}>} Sales)`	현재 선택과 연관된 2009년의 판매량을 반환하고 국가 Sweden과 연관된 수년 동안의 전체 데이터 집합을 추가합니다.
`Sum ({$<Country={"S*"}+{"*land"}>} Sales)`	s로 시작하거나 land로 끝나는 국가의 판매량을 반환합니다.

집합 수정자

집합 표현식은 계산 범위를 정의하는 데 사용됩니다. 집합 표현식의 중심 부분은 선택을 지정하는 집합 수정자입니다. 이는 사용자 선택 또는 집합 식별자의 선택을 수정하는 데 사용되며 결과는 계산을 위한 새 범위를 정의합니다.

집합 수정자는 하나 또는 여러 개의 필드 이름으로 구성되어 있는데, 각 필드 이름 다음에는 필드에 적용해야 하는 선택이 나옵니다. 수정자는 꺾쇠 괄호로 묶어야 합니다. < >

예:

- `Sum ( {$<Year = {2015}>} Sales )`
- `Count ( {1<Country = {Germany}>} distinct OrderID )`
- `Sum ( {$<Year = {2015}, Country = {Germany}>} Sales )`

요소 집합

요소 집합은 다음을 사용하여 정의할 수 있습니다.

- 값 목록
- 검색
- 다른 필드에 대한 참조
- 집합 함수

요소 집합 정의가 생략된 경우 집합 수정자는 이 필드의 모든 선택을 지웁니다. 예:

```
Sum( {$<Year = >} Sales )
```

예: 요소 집합을 기반으로 하는 집합 수정자에 대한 차트 표현식

예 - 차트 표현식

로드 스크립트

데이터 로드 편집기에서 다음 데이터를 인라인 로드로 로드하여 아래 차트 표현식 예를 만듭니다.

```
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
];
```

차트 표현식

다음 차트 표현식을 사용하여 Qlik Sense 시트에 테이블을 만듭니다.

테이블 - 요소 집합을 기반으로 한 집합 수정자

Country	**Sum (Sales)**	**Sum({1<Country={Belgium}>}Sales)**	**Sum ({1<Country={"*A*"}>}Sales)**	**Sum ({1<Country={"A*"}>}Sales)**	**Sum({1<Year={$(=Max (Year))}>}Sales)**
합계	1645397.3	360809.2	1284588.1	443238.88	788617.07
아르헨티나	206332.92	0	206332.92	206332.92	140037.89
오스트리아	236905.96	0	236905.96	236905.96	182739.87
벨기에	360809.2	360809.2	0	0	178042.33
브라질	176596.89	0	176596.89	0	2104.22
캐나다	142089.58	0	142089.58	0	40288.25
덴마크	152211.66	0	152211.66	0	106938.41
핀란드	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

설명

- 차원:
  - `Country`
- 측정값:

- Sum(Sales)
  집합 표현식 없는 합계 Sales입니다.
- Sum({1<Country={Belgium}>}Sales)
  Belgium을 선택한 다음 해당하는 Sales 합계를 구합니다.
- Sum({1<Country={"*A*"}>}Sales)
  A가 있는 모든 국가를 선택한 다음 해당하는 Sales 합계를 구합니다.
- Sum({1<Country={"A*"}>}Sales)
  A로 시작하는 모든 국가를 선택한 다음 해당하는 Sales 합계를 구합니다.
- Sum({1<Year={$(=Max(Year))}>}Sales)
  Max(Year)(2015)를 계산한 다음 해당하는 Sales 합계를 구합니다.

요소 집합 기반 집합 수정자

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

나열된 값

요소 집합의 가장 일반적인 예는 중괄호로 묶인 필드 값 목록을 기반으로 하는 것입니다. 예:

- {$<Country = {Canada, Germany, Singapore}>}
- {$<Year = {2015, 2016}>}

내부 중괄호는 요소 집합을 정의합니다. 개별 값은 쉼표로 구분됩니다.

따옴표 및 대/소문자 구분

값에 공백이나 특수 문자가 포함된 경우 값을 따옴표로 묶어야 합니다. 작은따옴표는 단일 필드 값과 대/소문자를 구분하는 리터럴 일치를 의미합니다. 큰따옴표는 하나 이상의 필드 값과 대/소문자를 구분하지 않는 일치를 의미합니다. 예:

- <Country = {'New Zealand'}>
  New Zealand만 일치합니다.
- <Country = {"New Zealand"}>
  New Zealand, NEW ZEALAND 및 new zealand가 일치합니다.

날짜는 따옴표로 묶어야 하며 해당 필드의 날짜 형식을 사용해야 합니다. 예:

- <ISO_Date = {'2021-12-31'}>
- <US_Date = {'12/31/2021'}>
- <UK_Date = {'31/12/2021'}>

큰따옴표는 대괄호나 억음 악센트 기호로 대체할 수 있습니다.

검색

검색을 통해 요소 집합을 만들 수도 있습니다. 예:

- <Country = {"C*"}>
- <Ingredient = {"*garlic*"}>
- <Year = {">2015"}>
- <Date = {">12/31/2015"}>

와일드카드는 텍스트 검색에 사용할 수 있습니다. 별표(*)는 임의의 수의 문자를 나타내고 물음표(?)는 단일 문자를 나타냅니다. 관계 연산자를 사용하여 숫자 검색을 정의할 수 있습니다.

검색에는 항상 큰따옴표를 사용해야 합니다. 검색은 대소문자를 구분하지 않습니다.

달러 기호 확장

요소 집합 내에서 계산을 사용하려면 달러 기호 확장이 필요합니다. 예를 들어, 가능한 마지막 연도만 보려면 다음을 사용할 수 있습니다.

```
<Year = {$(=Max(Year))}>
```

다른 필드에서 선택한 값

수정자는 다른 필드의 선택된 값을 기반으로 할 수 있습니다. 예:

```
<OrderDate = DeliveryDate>
```

이 수정자는 DeliveryDate에서 선택된 값을 가져와 OrderDate에 선택 내용으로 적용합니다. 고유 값(수백 개 이상)이 많을 경우 이 연산은 CPU에 부하를 주므로 피해야 합니다.

요소 집합 함수

요소 집합은 집합 함수 P()(가능한 값) 및 E()(제외된 값)을 기반으로 할 수도 있습니다.

예를 들어 Cap 제품이 판매된 국가를 선택하려면 다음을 사용할 수 있습니다.

```
<Country = P({1<Product={Cap}>} Country)>
```

마찬가지로 Cap 제품이 판매되지 않은 국가를 선택하려면 다음을 사용할 수 있습니다.

```
<Country = E({1<Product={Cap}>} Country)>
```

검색을 사용하는 집합 수정자

집합 수정자를 사용한 검색을 통해 요소 집합을 작성할 수 있습니다.

예:

- <Country = {"C*"}>
- <Year = {">2015"}>
- <Ingredient = {"*garlic*"}>

검색은 항상 큰따옴표, 대괄호 또는 억음 악센트 기호로 묶어야 합니다. 리터럴 문자열(작은따옴표)과 검색(큰따옴표)이 혼합된 목록을 사용할 수 있습니다. 예:

```
<Product = {'Nut', "*Bolt", Washer}>
```

텍스트 검색

와일드카드 및 기타 기호는 텍스트 검색에 사용할 수 있습니다.

- 별표(*)는 임의의 수의 문자를 나타냅니다.
- 물음표(?)는 단일 문자를 나타냅니다.
- 곡절 악센트(^)는 단어의 시작을 표시합니다.

예:

- <Country = {"C*", "*land"}>
  C로 시작하거나 land로 끝나는 모든 국가와 일치합니다.
- <Country = {"*^z*"}>
  New Zealand와 같이 z로 시작하는 단어가 있는 모든 국가와 일치합니다.

숫자 검색

다음 관계 연산자를 사용하여 숫자 검색을 수행할 수 있습니다. >, >=, <, <=

숫자 검색은 항상 이러한 연산자 중 하나로 시작됩니다. 예:

- <Year = {">2015"}>
  2016년 이후의 연도와 일치합니다.
- <Date = {">=1/1/2015<1/1/2016"}>
  2015년의 모든 날짜와 일치합니다. 두 날짜 사이의 시간 범위를 설명하는 구문에 유의하십시오. 날짜 형식은 해당 필드의 날짜 형식과 일치해야 합니다.

표현식 검색

표현식 검색을 사용하여 고급 검색을 수행할 수 있습니다. 그런 다음 검색 필드의 각 필드 값에 대해 집계가 평가됩니다. 검색 표현식이 true를 반환하는 모든 값이 선택됩니다.

표현식 검색은 항상 등호 기호로 시작됩니다. =

예:

```
<Customer = {"=Sum(Sales)>1000"}>
```

이렇게 하면 판매액이 1000보다 큰 모든 고객이 반환됩니다. Sum(Sales)는 현재 선택에서 계산됩니다. 즉, Product 필드와 같은 다른 필드에 선택 내용이 있는 경우 선택한 제품에 대해서만 판매 조건을 충족하는 고객이 반환됩니다.

조건이 선택과 무관하게 하려면 검색 문자열 내에서 집합 분석을 사용해야 합니다. 예:

```
<Customer = {"=Sum({1} Sales)>1000"}>
```

등호 뒤의 표현식은 부울 값으로 해석됩니다. 즉, 다른 것으로 평가되면 0이 아닌 숫자는 true로 해석되고 0과 문자열은 false로 해석됩니다.

Quotes

검색 문자열에 공백이나 특수 문자가 포함된 경우 인용 부호를 사용합니다. 작은따옴표는 단일 필드 값과 대/소문자를 구분하는 리터럴 일치를 의미합니다. 큰따옴표는 잠재적으로 여러 필드 값과 일치하는 대/소문자를 구분하지 않는 검색을 의미합니다.

예:

- `<Country = {'New Zealand'}>`
  `New Zealand`만 일치합니다.
- `<Country = {"New Zealand"}>`
  `New Zealand`, `NEW ZEALAND` 및 `new zealand`가 일치합니다.

큰따옴표는 대괄호나 억음 악센트 기호로 대체할 수 있습니다.

> *Qlik Sense의 이전 버전에서는 큰따옴표, 작은따옴표의 구분 없이 따옴표로 묶은 모든 문자열이 검색으로 처리되었습니다. 이전 버전과의 호환성을 유지하기 위해 Qlik Sense의 이전 버전으로 만든 앱은 계속해서 이전 버전에서와 마찬가지로 작동합니다. Qlik Sense November 2017 이상 버전으로 만든 앱은 두 가지 따옴표 유형 간의 차이를 구분합니다.*

예: 검색을 사용하는 집합 수정자에 대한 차트 표현식

예 - 차트 표현식

로드 스크립트

데이터 로드 편집기에서 다음 데이터를 인라인 로드로 로드하여 아래 차트 표현식 예를 만듭니다.

```
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];
```

예 1: 텍스트 검색을 사용하는 차트 표현식

다음 차트 표현식을 사용하여 Qlik Sense 시트에 테이블을 만듭니다.

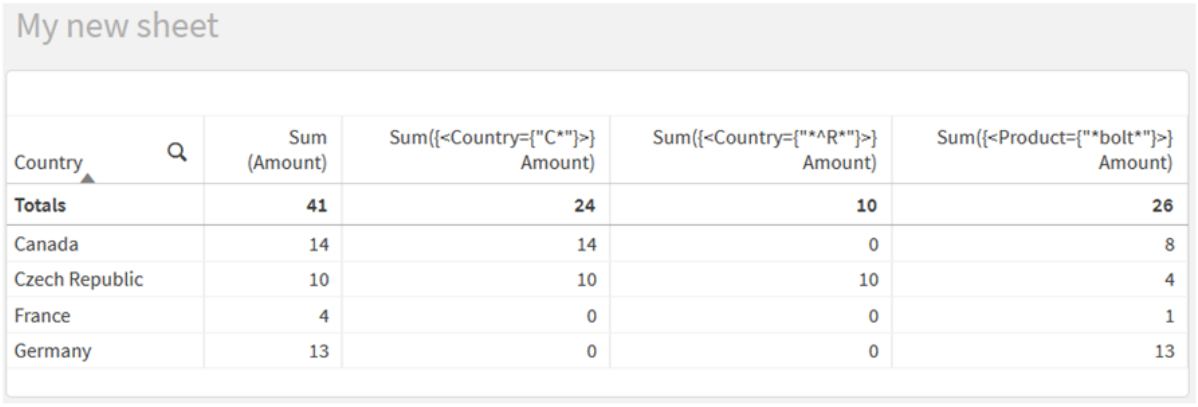
테이블 - 텍스트 검색을 사용하는 집합 수정자

Country	Sum (Amount)	Sum({<Country={"C*"}>} Amount)	Sum({<Country={"*^R*"}>} Amount)	Sum({<Product={"*bolt*"}>} Amount)
합계	**41**	**24**	**10**	**26**
캐나다	14	14	0	8
체코 공화국	10	10	10	4
France	4	0	0	1
Germany	13	0	0	13

설명

- 차원:
  - `Country`
- 측정값:
  - `Sum(Amount)`
    집합 표현식 없는 합계 `Amount`입니다.
  - `Sum({<Country={"C*"}>}Amount)`
    `C`로 시작하는 모든 국가(예: `Canada` 및 `Czech Republic`)의 합계 `Amount`입니다.
  - `Sum({<Country={"*^R*"}>}Amount)`
    `R`로 시작하는 단어가 있는 모든 국가(예: `Czech Republic`)의 합계 `Amount`입니다.
  - `Sum({<Product={"*bolt*"}>}Amount)`
    `bolt` 문자열을 포함하는 모든 제품(예: `Bolt` 및 `Anchor bolt`)의 합계 `Amount`입니다.

텍스트 검색을 사용하는 집합 수정자



예 2: 숫자 검색을 사용하는 차트 표현식

다음 차트 표현식을 사용하여 Qlik Sense 시트에 테이블을 만듭니다.

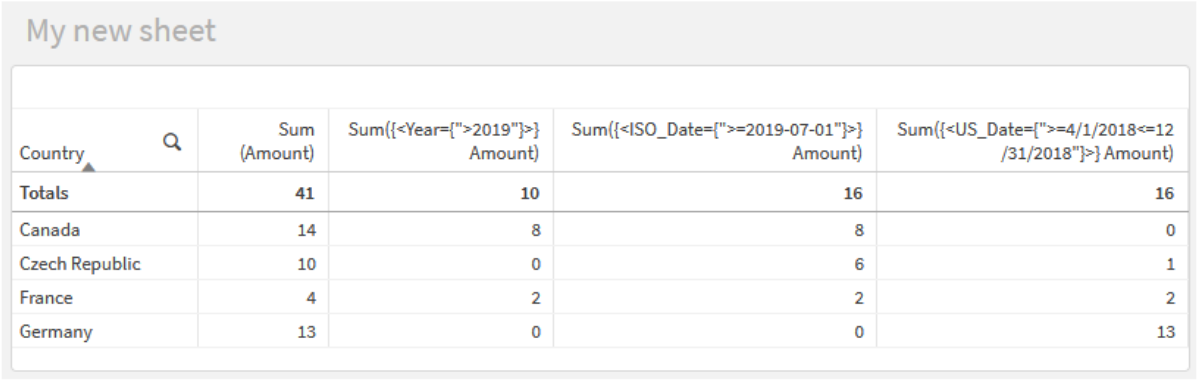
테이블 - 숫자 검색을 사용하는 집합 수정자

Country	Sum (Amount)	Sum({<Year={">2019"}>} Amount)	Sum({<ISO_Date={">=2019-07-01"}>} Amount)	Sum({<US_Date={">=4/1/2018<=12/31/2018"}>} Amount)
합계	**41**	**10**	**16**	**16**
캐나다	14	8	8	0
체코 공화국	10	0	6	1
France	4	2	2	2
Germany	13	0	0	13

설명

- 차원:
  - `Country`
- 측정값:
  - `Sum(Amount)`
    집합 표현식 없는 합계 `Amount`입니다.
  - `Sum({<Year={">2019"}>}Amount)`
    2019 이후의 모든 연도에 대한 합계 `Amount`입니다.
  - `Sum({<ISO_Date={">=2019-07-01"}>}Amount)`
    2019-07-01 또는 그 이후의 모든 날짜에 대한 합계 `Amount`입니다. 검색의 날짜 형식은 필드 형식과 일치해야 합니다.
  - `Sum({<US_Date={">=4/1/2018<=12/31/2018"}>}Amount)`
    시작 날짜와 종료 날짜를 포함하여 `4/1/2018`부터 `12/31/2018`까지의 모든 날짜에 대한 합계 `Amount`입니다. 검색의 날짜 형식은 필드 형식과 일치해야 합니다.

숫자 검색을 사용하는 집합 수정자



예 3: 표현식 검색을 사용하는 차트 표현식

다음 차트 표현식을 사용하여 Qlik Sense 시트에 테이블을 만듭니다.

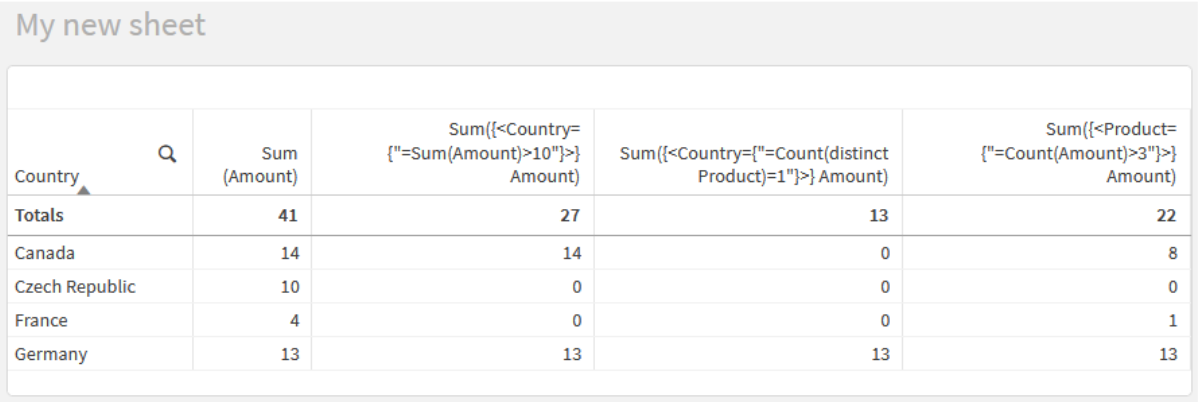
Table - Set modifiers with expression searches

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

설명

- 차원:
  - `Country`
- 측정값:
  - `Sum(Amount)`
    집합 표현식 없는 합계 `Amount`입니다.
  - `Sum({<Country={"=Sum(Amount)>10"}>}Amount)`
    집계된 `Amount` 합계가 10보다 큰 모든 국가의 합계 `Amount`입니다.
  - `Sum({<Country={"=Count(distinct Product)=1"}>}Amount)`
    정확히 하나의 고유한 제품과 연관된 모든 국가의 합계 `Amount`입니다.
  - `Sum({<Product={"=Count(Amount)>3"}>}Amount)`
    데이터에 4개 이상의 거래가 있는 모든 국가의 합계 `Amount`입니다.

표현식 검색을 사용하는 집합 수정자



예	결과
sum({$–1<Product = {“*Internal*”, “*Domestic*”}>} Sales)	제품 이름에 'Internal' 또는 'Domestic' 문자열이 있는 제품과 관련된 거래를 제외하고 현재 선택의 판매량을 반환합니다.

예	결과
sum({$<Customer = {"=Sum({1<Year = {2007}>} Sales) > 1000000"}>} Sales)	'Customer' 필드에 2007년 총 판매량이 1000000 이상인 고객만 포함되도록 새로 선택하여 현재 선택의 판매량을 반환합니다.

달러 기호 확장을 사용하는 집합 수정자

달러 기호 확장은 표현식이 구문 분석 및 평가되기 전에 계산되는 구문입니다. 그런 다음 결과가 $(...) 대신 표현식에 주입됩니다. 그런 다음 달러 기호 확장의 결과를 사용하여 표현식을 계산합니다.

식 편집기는 달러 기호 확장 미리보기를 표시하므로 달러 기호 확장이 무엇을 평가하는지 확인할 수 있습니다.

식 편집기의 달러 기호 확장 미리보기

요소 집합 내에서 계산을 사용하려면 달러 기호 확장을 사용합니다.

예를 들어, 가능한 마지막 연도만 보고 싶다면 다음 구성을 사용할 수 있습니다.

```
<Year = {$(=Max(Year))}>
```

Max(Year)가 먼저 계산되고 결과가 $(...) 대신 표현식에 주입됩니다.

달러 확장 후의 결과는 다음과 같은 표현식이 됩니다.

```
<Year = {2021}>
```

달러 기호 확장 안의 표현식은 현재 선택을 기반으로 계산됩니다. 즉, 다른 필드에 선택 내용이 있는 경우 표현식의 결과가 영향을 받습니다.

계산이 선택과 무관하게 하려면 달러 기호 확장 내에서 집합 분석을 사용합니다. 예:

```
<Year = {$(=Max({1} Year))}>
```

문자열

달러 기호 확장으로 인해 문자열이 생성되도록 하려면 일반적인 인용 규칙이 적용됩니다. 예:

```
<Country = {'$(=FirstSortedValue(Country,Date)'}>
```

달러 확장 후의 결과는 다음과 같은 표현식이 됩니다.

```
<Country = {'New Zealand'}>
```

인용 부호를 사용하지 않으면 구문 오류가 발생합니다.

숫자

달러 기호 확장으로 인해 숫자가 표시되도록 하려면 확장이 필드와 동일한 형식을 가져야 합니다. 즉, 서식 지정 함수에서 표현식을 래핑해야 하는 경우가 있습니다.

예:

```
<Amount = {$(=Num(Max(Amount), '###0.00'))}>
```

달러 확장 후의 결과는 다음과 같은 표현식이 됩니다.

```
<Amount = {12362.00}>
```

해시를 사용하여 확장이 항상 소수점을 사용하고 천 단위 구분 기호를 사용하지 않도록 합니다. 예:

```
<Amount = {$(#=Max(Amount))}>
```

날짜

달러 기호 확장으로 인해 날짜가 표시되도록 하려면 확장이 올바른 형식이어야 합니다. 즉, 서식 지정 함수에서 표현식을 래핑해야 하는 경우가 있습니다.

예:

```
<Date = {'$(=Date(Max(Date)))'}>
```

달러 확장 후의 결과는 다음과 같은 표현식이 됩니다.

```
<Date = {'12/31/2015'}>
```

문자열과 마찬가지로 올바른 따옴표를 사용해야 합니다.

일반적인 사용 사례는 계산을 지난 달(또는 연도)로 제한하려는 것입니다. 그런 다음 `AddMonths()` 함수와 함께 숫자 검색을 사용할 수 있습니다.

예:

```
<Date = {">=$(=AddMonths(Today(),-1))"}>
```

달러 확장 후의 결과는 다음과 같은 표현식이 됩니다.

```
<Date = {">=9/31/2021"}>
```

이 함수는 지난 달에 발생한 모든 이벤트를 선택합니다.

예: 달러 기호 확장을 사용하는 집합 수정자에 대한 차트 표현식

예 - 차트 표현식

로드 스크립트

데이터 로드 편집기에서 다음 데이터를 인라인 로드로 로드하여 아래 차트 표현식 예를 만듭니다.

```
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];
```

달러 기호 확장을 사용하는 차트 표현식

다음 차트 표현식을 사용하여 Qlik Sense 시트에 테이블을 만듭니다.

테이블 - 달러 기호 확장을 사용하는 집합 수정자

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)
합계	**41**	**1**	**6**	**1**
캐나다	14	0	6	0
체코 공화국	10	0	0	0
France	4	1	0	1
Germany	13	0	0	0

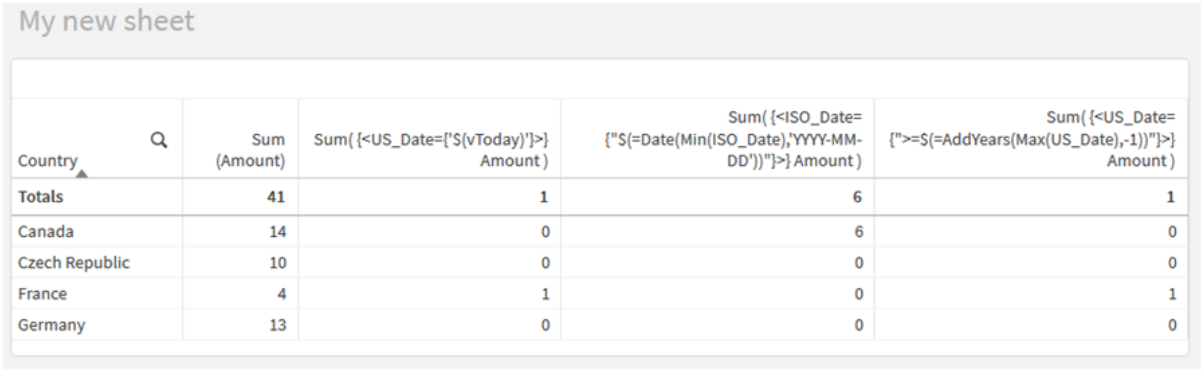
설명

- 차원:
  - `Country`
- 측정값:
  - `Sum(Amount)`
    집합 표현식 없는 합계 `Amount`입니다.
  - `Sum({<US_Date={'$(vToday)'}>}Amount)`
    `Amount`가 `US_Date` 변수`vToday`와 동일한 모든 레코드의 합계입니다.
  - `Sum({<ISO_Date={"$(=Date(Min(ISO_Date),'YYYY-MM-DD'))"}>}Amount)`

ISO_Date가 첫 번째(가장 작은) 가능한 ISO_Date와 동일한 모든 레코드의 합계 Amount입니다. 날짜 형식이 필드 형식과 일치하도록 하려면 Date() 함수가 필요합니다.

- Sum({<US_Date={">=$(=AddYears(Max(US_Date),-1))"}>}Amount)
  마지막(가장 큰) 가능한 US_Date의 1년 전 날짜 또는 그 이후에 US_Date가 있는 모든 레코드의 합계 Amount입니다. AddYears() 함수는 변수 DateFormat에 지정된 형식으로 날짜를 반환하며 이는 필드 US_Date 형식과 일치해야 합니다.

달러 기호 확장을 사용하는 집합 수정자


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

예	결과
sum({$<Year = {$(#vLastYear)}>} Sales)	현재 선택을 기준으로 이전 연도의 판매량을 반환합니다. 여기에서 관련 연도를 포함하는 vLastYear 변수는 달러 기호 확장에 사용됩니다.
sum({$<Year = {$(#=Only(Year)-1)}>} Sales)	현재 선택을 기준으로 이전 연도의 판매량을 반환합니다. 여기에서 달러 기호 확장은 이전 연도를 계산하는 데 사용합니다.

집합 연산자를 사용하는 집합 수정자

집합 연산자는 다른 요소 집합을 포함, 제외 또는 교차하는 데 사용됩니다. 이 연산자는 요소 집합을 정의하기 위해 다른 방법을 결합합니다.

이 연산자는 집합 식별자에 사용되는 것과 동일합니다.

연산자

연산자	설명
+	합집합. 이 2항 연산은 두 집합 피연산자 중 하나에라도 속하는 레코드 또는 요소로 구성된 집합을 반환합니다.
-	차집합. 이 2항 연산은 두 집합 피연산자 중 첫 번째 피연산자에 속하지만 다른 피연산자에는 속하지 않는 레코드 또는 요소로 구성된 집합을 반환합니다. 또한 단항 연산자로 사용된 경우 여집합을 반환합니다.
*	교집합. 이 2항 연산은 두 집합 피연산자 모두에 속하는 레코드 또는 요소로 구성된 집합을 반환합니다.

연산자	설명
/	대칭차(XOR). 이 2항 연산은 두 집합 피연산자 중 하나에 속하지만 두 집합 피연산자 모두에 속하지 않는 레코드 또는 요소로 구성된 집합을 반환합니다.

예를 들어 다음 두 수정자는 동일한 필드 값 집합을 정의합니다.

- <Year = {1997, "20*"}>
- <Year = {1997} + {"20*"}>

두 표현식 모두 1997과 20로 시작하는 연도를 선택합니다. 즉, 이는 두 조건의 합집합입니다.

집합 연산자는 또한 더 복잡한 정의를 허용합니다. 예:

```
<Year = {1997, "20*"} - {2000}>
```

이 표현식은 위와 같은 연도를 선택하지만 추가로 2000년은 제외합니다.

예: 집합 연산자를 사용하는 집합 수정자에 대한 차트 표현식

예 - 차트 표현식

로드 스크립트

데이터 로드 편집기에서 다음 데이터를 인라인 로드로 로드하여 아래 차트 표현식 예를 만듭니다.

```
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];
```

차트 표현식

다음 차트 표현식을 사용하여 Qlik Sense 시트에 테이블을 만듭니다.

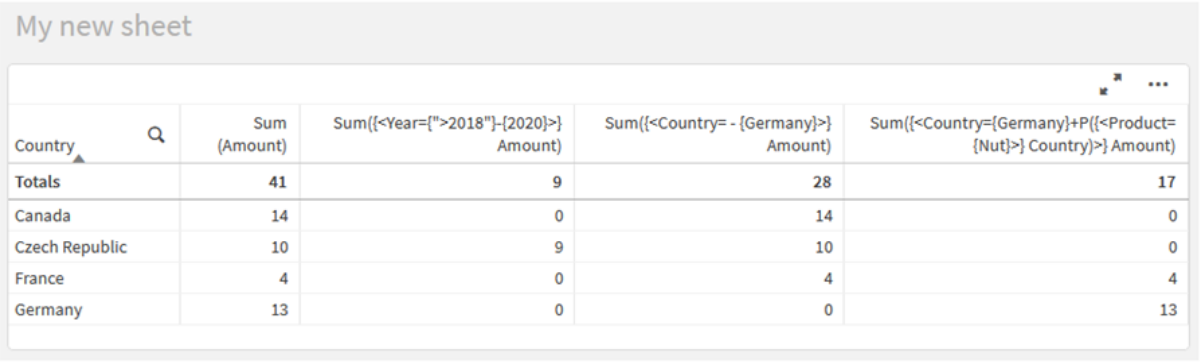
테이블 - 집합 연산자를 사용하는 집합 수정자

Country	Sum (Amount)	Sum({<Year={">2018"}-{2020}>} Amount)	Sum({<Country=-{Germany}>} Amount)	Sum({<Country={Germany}+P({<Product={Nut}>}Country)>} Amount)
합계	**41**	**9**	**28**	**17**
캐나다	14	0	14	0
체코 공화국	10	9	10	0
France	4	0	4	4
Germany	13	0	0	13

설명

- 차원:
  - `Country`
- 측정값:
  - `Sum(Amount)`
    집합 표현식 없는 합계 `Amount`입니다.
  - `Sum({<Year={">2018"}-{2020}>}Amount)`
    2020을 제외한 2018 이후의 모든 연도에 대한 합계 `Amount`입니다.
  - `Sum({<Country=-{Germany}>}Amount)`
    `Germany`를 제외한 모든 국가의 합계 `Amount`입니다. 단항 제외 연산자에 유의하십시오.
  - `Sum({<Country={Germany}+P({<Product={Nut}>}Country)>}Amount)`
    `Germany`와 제품 `Nut`와 연관된 모든 국가의 합계 `Amount`입니다.

집합 연산자를 사용하는 집합 수정자



예	결과
sum({$<Product = Product + {OurProduct1} – {OurProduct2} >} Sales)	선택된 제품 목록에서 "OurProduct1" 제품을 추가하고 "OurProduct2"를 제거한 후 현재 선택의 판매량을 반환합니다.

예	결과
sum({$<Year = Year + ({“20*”,1997} – {2000}) >} Sales)	“Year” 필드에서 1997년 및 “20”으로 시작하는 모든 연도(2000년 제외)를 추가로 선택하고 현재 선택의 판매량을 반환합니다. 현재 선택에 2000이 포함된 경우 수정 후 포함됩니다.
sum({$<Year = (Year + {“20*”,1997}) – {2000} >} Sales)	위와 거의 같은 내용을 반환하지만, 현재 선택에 원래 포함된 경우에도 2000이 제외됩니다. 이 예는 경우에 따라 괄호를 사용하여 우선 순위의 순서를 정의하는 것이 얼마나 중요한지를 보여줍니다.
sum({$<Year = {“*”} – {2000}, Product = {“*bearing*”} >} Sales)	"Year"에 2000을 제외한 모든 연도를 새로 선택하고 ‘bearing’ 문자열을 포함한 제품에만 해당하는 현재 선택의 판매량을 반환합니다.

암시적 집합 연산자를 사용하는 집합 수정자

집합 수정자에서 선택을 작성하는 표준 방법은 등호를 사용하는 것입니다. 예:

```
Year = {">2015"}
```

집합 수정자의 등호 오른쪽에 있는 표현식을 요소 집합이라고 합니다. 이는 별개의 필드 값 집합, 즉 선택을 정의합니다.

이 표기법은 필드의 현재 선택을 무시하고 새 선택을 정의합니다. 따라서 집합 식별자에 이 필드의 선택이 포함되어 있으면 이전 선택이 요소 집합의 항목으로 대체됩니다.

필드의 현재 선택을 기준으로 선택하려면 다른 표현식을 사용해야 합니다.

예를 들어, 이전 선택을 유지하고 연도가 2015년 이후여야 한다는 요구 사항을 추가하려면 다음을 작성할 수 있습니다.

```
Year = Year * {">2015"}
```

별표는 교집합을 정의하는 집합 연산자이므로 Year에서 현재 선택과 연도가 2015 이후여야 하는 추가 요구 사항 간의 교집합을 얻을 수 있습니다. 이를 작성하는 다른 방법은 다음과 같습니다.

```
Year *= {">2015"}
```

즉, 할당 연산자(*=)는 교집합을 암시적으로 정의합니다.

마찬가지로, 암시적 합집합, 제외 및 대칭차는 다음을 사용하여 정의할 수 있습니다. +=, -=, /=

예: 암시적 집합 연산자를 사용하는 집합 수정자에 대한 차트 표현식

예 - 차트 표현식

로드 스크립트

데이터 로드 편집기에서 다음 데이터를 인라인 로드로 로드하여 아래 차트 표현식 예를 만듭니다.

```
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];
```

암시적 집합 연산자를 사용하는 차트 표현식

다음 차트 표현식을 사용하여 Qlik Sense 시트에 테이블을 만듭니다.

국가 목록에서 `Canada` 및 `Czech Republic`을 선택합니다.

테이블 - 암시적 집합 연산자를 사용하는 차트 표현식

Country	Sum (Amount)	Sum({<Country*= {Canada}>} Amount)	Sum({<Country-= {Canada}>} Amount)	Sum({<Country+= {France}>} Amount)
합계	**24**	**14**	**10**	**28**
캐나다	14	14	0	14
체코 공화국	10	0	10	10
France	0	0	0	4

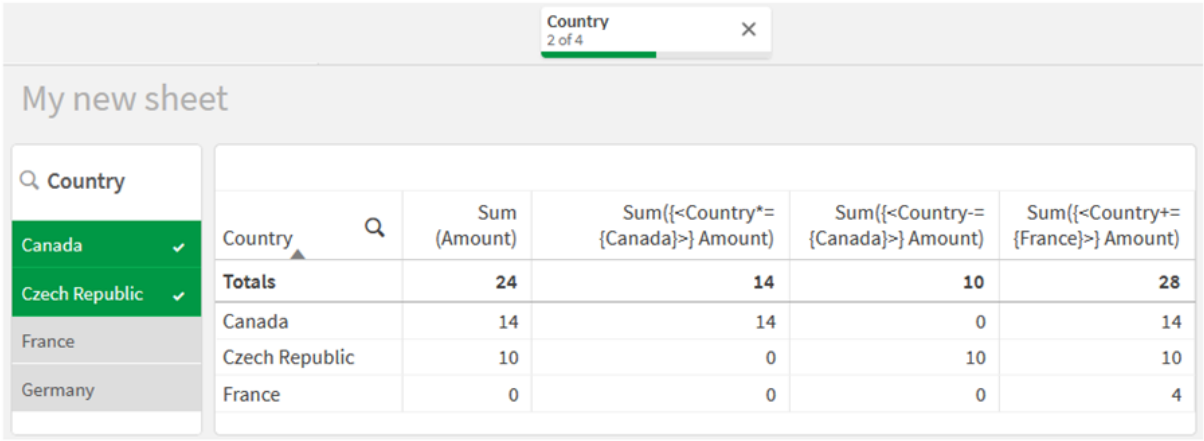
설명

- 차원:
  - `Country`
- 측정값:
  - `Sum(Amount)`
    현재 선택에 대한 합계 `Amount`입니다. `Canada` 및 `Czech Republic`만 `0`이 아닌 값을 가집니다.
  - `Sum({<Country*={Canada}>}Amount)`
    `Country`가 `Canada`여야 한다는 요구 사항과 교차하는 현재 선택의 합계 `Amount`입니다. `Canada`가 사용자 선택의 일부가 아닌 경우 집합 표현식은 빈 집합을 반환하고 열은 모든 행에서 `0`을 갖습니다.
  - `Sum({<Country-={Canada}>}Amount)`

현재 선택의 합계 Amount이지만 먼저 Country 선택에서 Canada를 제외합니다. Canada가 사용자 선택의 일부가 아닌 경우 집합 표현식은 숫자를 변경하지 않습니다.

- Sum({<Country+={France}>}Amount)
  현재 선택에 대한 합계 Amount이지만 먼저 Country 선택에 France를 추가합니다. France가 이미 사용자 선택의 일부인 경우 집합 표현식은 숫자를 변경하지 않습니다.

암시적 집합 연산자를 사용하는 집합 수정자



예	결과
sum({$<Product += {OurProduct1, OurProduct2} >} Sales)	현재 선택의 판매량을 반환하되, 암시적 합집합을 사용하여 'OurProduct1' 및 'OurProduct2' 제품을 선택된 제품 목록에 추가합니다.
sum({$<Year += {“20*”,1997} – {2000} >} Sales)	현재 선택의 판매량을 반환하되, 암시적 합집합을 사용하여 1997년 및 “20”으로 시작하는 모든 연도(2000년 제외)를 선택에 추가합니다. 현재 선택에 2000이 포함된 경우 수정 후 포함됩니다. <Year=Year + ({“20*”,1997}-{2000})>와 같습니다.
sum({$<Product *= {OurProduct1} >} Sales)	현재 선택된 제품과 OurProduct1 제품의 교집합에 해당하는 현재 선택의 판매량을 반환합니다.

집합 함수를 사용하는 집합 수정자

중첩된 집합 정의를 사용하여 필드 값 집합을 정의해야 하는 경우가 있습니다. 예를 들어 제품을 선택하지 않고 특정 제품을 구매한 모든 고객을 선택할 수 있습니다.

이러한 경우 요소 집합 함수 P() 및 E()를 사용합니다. 필드의 가능한 값과 제외된 값의 요소 집합을 각각 반환합니다. 대괄호 안에는 해당 필드와 범위를 정의하는 집합 표현식을 지정할 수 있습니다. 예:

```
P({1<Year = {2021}>} Customer)
```

그러면 2021년에 거래가 있었던 고객 집합이 반환됩니다. 그런 다음 집합 수정자에서 이를 사용할 수 있습니다. 예:

```
Sum({<Customer = P({1<Year = {2021}>} Customer)>} Amount)
```

이 집합 표현식은 이러한 고객을 선택하지만 선택을 2021년으로 제한하지 않습니다.

이러한 함수는 다른 표현식에는 사용할 수 없습니다.

또한 요소 집합 함수 내에서는 자연 집합만 사용할 수 있습니다. 즉, 간단한 선택으로 정의할 수 있는 레코드 집합입니다.

예를 들어 {1-$}로 지정된 집합은 선택을 통해 정의하지 못할 수도 있으므로 자연 집합이 아닙니다. 자연이 아닌 집합에서 이러한 함수를 사용하면 예기치 않은 결과가 반환됩니다.

예: 집합 함수를 사용하는 집합 수정자에 대한 차트 표현식

예 - 차트 표현식

로드 스크립트

데이터 로드 편집기에서 다음 데이터를 인라인 로드로 로드하여 아래 차트 표현식 예를 만듭니다.

```
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];
```

차트 표현식

다음 차트 표현식을 사용하여 Qlik Sense 시트에 테이블을 만듭니다.

테이블 - 집합 함수를 사용하는집합 수정자

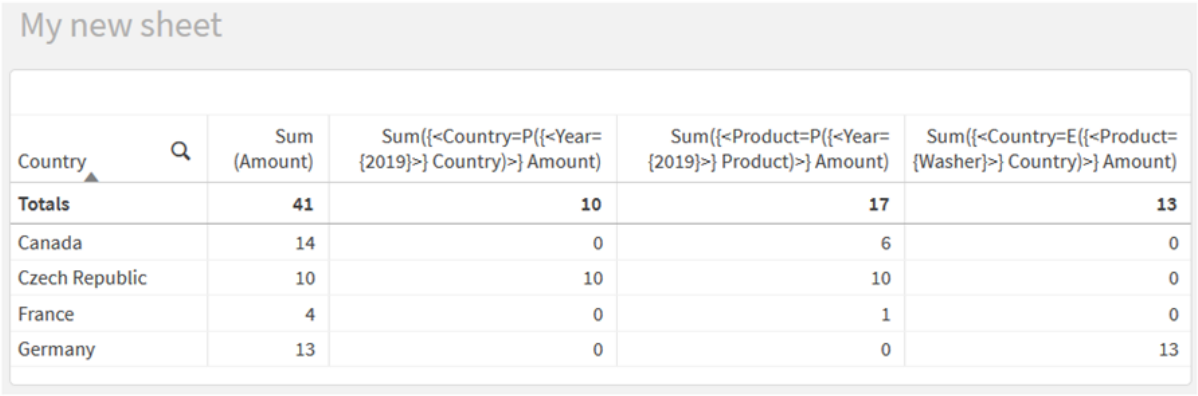
Country	Sum (Amount)	Sum({<Country=P ({<Year= {2019}>}Country)>} Amount)	Sum({<Product=P ({<Year= {2019}>}Product)>} Amount)	Sum({<Country=E ({<Product= {Washer}>}Country)>} Amount)
합계	**41**	**10**	**17**	**13**
캐나다	14	0	6	0
체코 공화	10	10	10	0

Country	Sum (Amount)	Sum({<Country=P({<Year={2019}>}Country)>} Amount)	Sum({<Product=P({<Year={2019}>}Product)>} Amount)	Sum({<Country=E({<Product={Washer}>}Country)>} Amount)
합계	**41**	**10**	**17**	**13**
국				
France	4	0	1	0
Germany	13	0	0	13

설명

- 차원:
  - `Country`
- 측정값:
  - `Sum(Amount)`
    집합 표현식 없는 합계 `Amount`입니다.
  - `Sum({<Country=P({<Year={2019}>} Country)>} Amount)`
    연도 `2019`와 연관된 국가의 합계 `Amount`입니다. 그러나 계산을 `2019`로 제한하지 않습니다.
  - `Sum({<Product=P({<Year={2019}>} Product)>} Amount)`
    연도 `2019`와 연관된 제품의 합계 `Amount`입니다. 그러나 계산을 `2019`로 제한하지 않습니다.
  - `Sum({<Country=E({<Product={Washer}>} Country)>} Amount)`
    제품 `Washer`와 연관되지 않은 국가의 합계`Amount`입니다.

집합 함수를 사용하는 집합 수정자



예	결과
sum({$<Customer = P ({1<Product= {'Shoe'}>} Customer)>} Sales)	'Shoe' 제품을 구매한 적이 있는 고객에 한해 현재 선택의 판매량을 반환합니다. 여기서 요소 함수 P( )는 Product 필드의 'Shoe' 선택으로 유추된 사용 가능한 고객 목록을 반환합니다.
sum({$<Customer = P ({1<Product= {'Shoe'}>})>} Sales)	위와 같습니다. 요소 함수의 필드가 생략된 경우 이 함수는 외부 할당에서 지정한 필드의 사용 가능한 값을 반환합니다.
sum({$<Customer = P ({1<Product= {'Shoe'}>} Supplier)>} Sales)	'Shoe' 제품을 공급한 적이 있는 고객에 한해 현재 선택의 판매량을 반환합니다. 즉, 고객도 공급자입니다. 여기서 요소 함수 P( )는 Product 필드의 'Shoe' 선택으로 유추된 사용 가능한 공급자 목록을 반환합니다. 이때 공급자 목록은 Customer 필드에서 선택 항목으로 사용됩니다.
sum({$<Customer = E ({1<Product= {'Shoe'}>})>} Sales)	'Shoe' 제품을 구매한 적이 없는 고객에 한해 현재 선택의 판매량을 반환합니다. 여기서 요소 함수 E( )는 Product 필드에서 'Shoe' 선택에 의해 제외된 고객 목록을 반환합니다.

내부 및 외부 집합 표현식

집합 표현식은 집계 함수 내부 및 외부에서 사용할 수 있으며 중괄호로 묶입니다.

집계 함수 내에서 집합 표현식을 사용하면 다음과 같이 보일 수 있습니다.

내부 집합 표현식

```
Sum( {$<Year={2021}>} Sales )
```

표현식에 여러 집계가 있고 모든 집계 함수에서 동일한 집합 표현식을 작성하지 않으려면 집계 함수 외부에서 집합 표현식을 사용합니다.

외부 집합 표현식을 사용하는 경우 범위 시작 부분에 배치해야 합니다.

외부 집합 표현식

```
{<Year={2021}>} Sum(Sales) / Count(distinct Customer)
```

집계 함수 외부에서 집합 표현식을 사용하는 경우 기존 마스터 측정값에도 적용할 수 있습니다.

마스터 측정값에 적용된 외부 집합 표현식

```
{<Year={2021}>} [Master Measure]
```

집계 함수 외부에서 사용되는 집합 표현식은 전체 표현식에 영향을 미칩니다. 단, 괄호로 묶인 경우에는 괄호가 범위를 정의합니다. 아래의 어휘 범위 지정 예에서 집합 표현식은 괄호 안의 집계에만 적용됩니다.

어휘 범위 지정

```
( {<Year={2021}>} Sum(Amount) / Count(distinct Customer) ) - Avg(CustomerSales)
```

규칙

어휘 범위
집합 표현식은 괄호로 묶이지 않는 한 전체 표현식에 영향을 줍니다. 이 경우 괄호는 어휘 범위를 정의합니다.

위치
집합 표현식은 어휘 범위의 시작 부분에 위치해야 합니다.

컨텍스트
컨텍스트는 표현식과 관련된 선택 항목입니다. 일반적으로 컨텍스트는 항상 현재 선택 항목의 기본 상태였습니다. 그러나 개체가 대체 상태로 설정된 경우 컨텍스트는 현재 선택 항목의 대체 상태입니다.

외부 집합 표현식의 형태로 컨텍스트를 정의할 수도 있습니다.

상속
내부 집합 표현식은 외부 집합 표현식보다 우선합니다. 내부 집합 표현식에 집합 식별자가 포함되어 있으면 컨텍스트를 바꿉니다. 그렇지 않으면 컨텍스트와 집합 표현식이 병합됩니다.

- `{$<SetExpression>}` - 외부 집합 표현식을 재정의합니다.
- `{<SetExpression>}` - 외부 집합 표현식과 병합됩니다.

요소 집합 할당
요소 집합 할당은 두 선택 항목이 병합되는 방법을 결정합니다. 일반 등호를 사용하는 경우 내부 집합 표현식의 선택 항목이 우선합니다. 그렇지 않으면 암시적 집합 연산자가 사용됩니다.

- `{<Field={value}>}` - 이 내부 선택 항목은 "`Field`"의 모든 외부 선택 항목을 바꿉니다.
- `{<Field+={value}>}` - 이 내부 선택 항목은 통합 연산자를 사용하여 "`Field`"의 외부 선택 항목과 병합됩니다.
- `{<Field*={value}>}` - 이 내부 선택 항목은 교차 연산자를 사용하여 "`Field`"의 외부 선택 항목과 병합됩니다.

여러 단계의 상속
상속은 여러 단계에서 발생할 수 있습니다. 예:

- 현재 선택 항목 → `Sum(Amount)`
  집계 함수는 현재 선택 항목인 컨텍스트를 사용합니다.
- 현재 선택 항목 → `{<Set1>} Sum(Amount)`
  `Set1`은 현재 선택 항목에서 상속되고 결과는 집계 함수의 컨텍스트가 됩니다.
- 현재 선택 항목 → `{<Set1>} ({<Set2>} Sum(Amount))`
  `Set2`는 `Set1`에서 상속되며, 이는 차례로 현재 선택 항목에서 상속되며 결과는 집계 함수의 컨텍스트가 됩니다.

Aggr() 함수

Aggr() 함수는 두 개의 독립적인 집계가 있는 중첩 집계를 만듭니다. 아래 예에서 Count()는 Dim의 각 값에 대해 계산되고 결과 배열은 Sum() 함수를 사용하여 집계됩니다.

```
Sum(Aggr(Count(X),Dim))
```

Count()는 내부 집계이고 Sum()은 외부 집계입니다.

- 내부 집계는 외부 집계에서 컨텍스트를 상속하지 않습니다.
- 내부 집계는 집합 표현식을 포함할 수 있는 Aggr() 함수에서 컨텍스트를 상속합니다.
- Aggr() 함수와 외부 집계 함수는 모두 외부 집합 표현식에서 컨텍스트를 상속합니다.

자습서 - 집합 표현식 만들기

데이터 분석을 지원하기 위해 Qlik Sense에서 집합 표현식을 작성할 수 있습니다. 이러한 컨텍스트에서 분석은 종종 집합 분석이라고 합니다. 집합 분석은 앱의 현재 선택 항목으로 정의된 레코드 집합과 다른 범위를 정의하는 방법을 제공합니다.

학습 내용

이 자습서에서는 집합 수정자, 식별자 및 연산자를 사용하여 집합 표현식을 작성하기 위한 데이터 및 차트 표현식을 제공합니다.

이 자습서의 대상

이 자습서는 스크립트 편집기 및 차트 표현식으로 작업하는 데 익숙한 앱 개발자를 위한 것입니다.

준비 사항

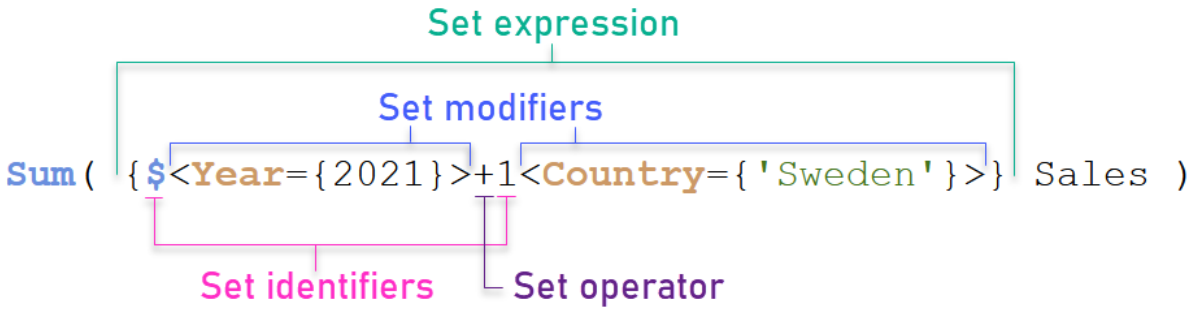
데이터를 로드하고 앱을 만들 수 있는 Qlik Sense Enterprise Professional 액세스 할당입니다.

- 집합 분석 1부: 초보자를 위한 소개
- 집합 분석 2부

집합 표현식의 요소

집합 표현식을 Sum(), Max(), Min(), Avg() 또는 Count() 등 집계 함수로 묶습니다. 집합 표현식은 요소라고 하는 핵심 구성 요소로 구성됩니다. 이러한 요소는 집합 수정자, 식별자 및 연산자입니다.

집합 표현식의 요소



예를 들어 위의 집합 표현식은 집계 `Sum(Sales)`에서 작성됩니다. 집합 표현식은 외부 중괄호로 묶입니다. `{ }`

표현식의 첫 번째 피연산자는 다음과 같습니다. `$<Year={2021}>`

이 피연산자는 현재 선택 항목에 대한 2021년의 판매량을 반환합니다. 수정자(`<Year={2021}>`)는 2021년의 선택을 포함합니다. `$` 집합 식별자는 집합 표현식이 현재 선택을 기반으로 함을 나타냅니다.

표현식의 두 번째 피연산자는 다음과 같습니다. `1<Country={'Sweden'}>`

이 피연산자는 Sweden의 Sales를 반환합니다. 수정자(`<Country={'Sweden'}>`)는 국가 Sweden 선택을 포함합니다. `1` 집합 식별자는 앱에서 선택한 항목이 무시됨을 나타냅니다.

마지막으로 `+` 집합 연산자는 표현식이 두 집합 피연산자 중 하나에 속하는 레코드로 구성된 집합을 반환함을 나타냅니다.

집합 표현식 만들기 자습서

이 자습서에 표시된 집합 표현식을 만들려면 다음 절차를 완료합니다.

새 앱 만들기 및 데이터 로드

다음과 같이 하십시오.

1. 새 앱을 만듭니다.
2. **스크립트 편집기**를 클릭합니다. 또는 탐색 막대에서 **준비** > **데이터 로드 편집기**를 클릭합니다.
3. **데이터 로드 편집기**에서 새 섹션을 만듭니다.
4. 다음 데이터를 복사하여 새 섹션에 붙여넣습니다. *집합 표현식 자습서 데이터 (page 308)*
5. **데이터 로드**를 클릭합니다. 데이터가 인라인 로드로 로드됩니다.

수정자를 사용하여 집합 표현식 만들기

집합 수정자는 하나 또는 여러 개의 필드 이름으로 구성되어 있는데, 각 필드 이름 다음에는 필드에 적용해야 하는 선택이 나옵니다. 수정자는 꺾쇠 괄호로 묶어야 합니다. 예를 들어, 이 집합 표현식에서:

```
Sum ( {<Year = {2015}>} Sales )
```

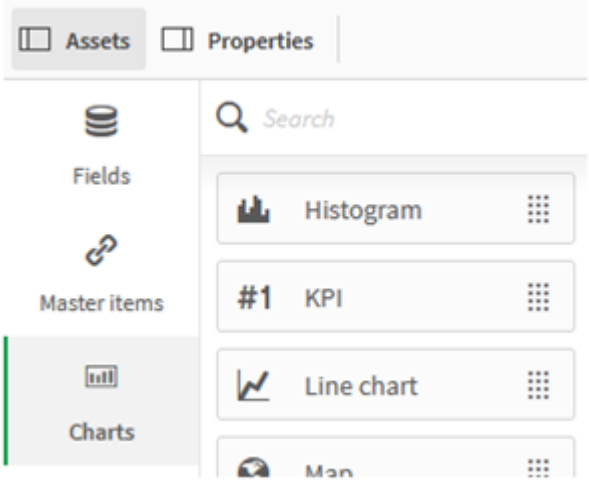
수정자:

```
<Year = {2015}>
```

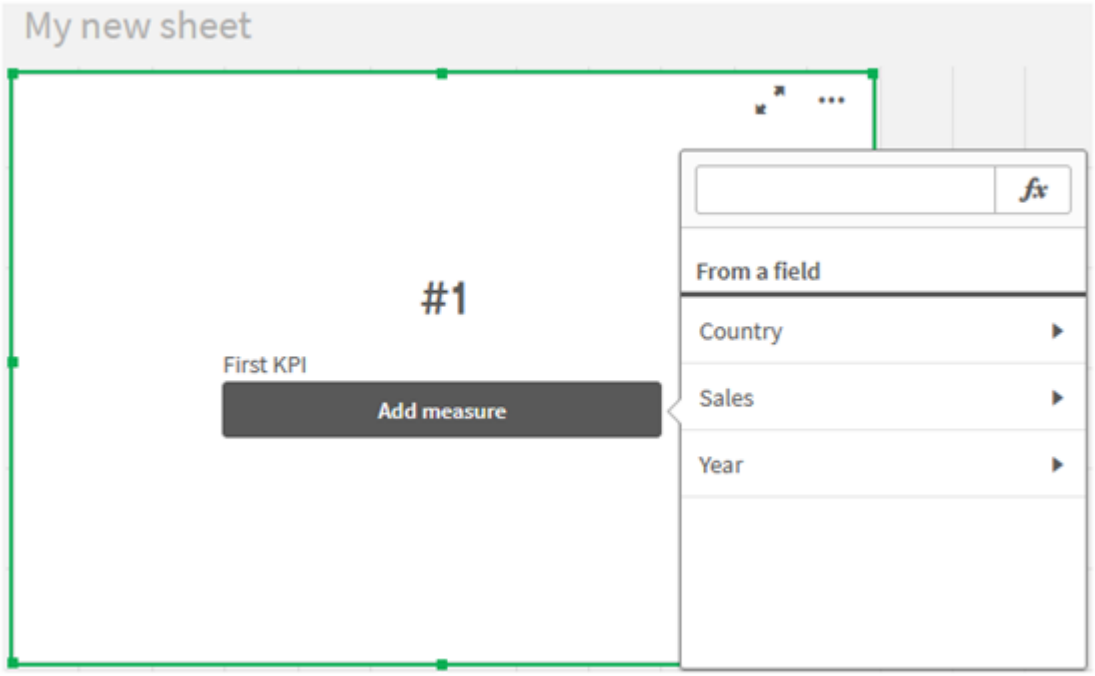
이 수정자는 2015년의 데이터가 선택되도록 지정합니다. 수정자가 묶인 중괄호는 집합 표현식을 나타냅니다.

다음과 같이 하십시오.

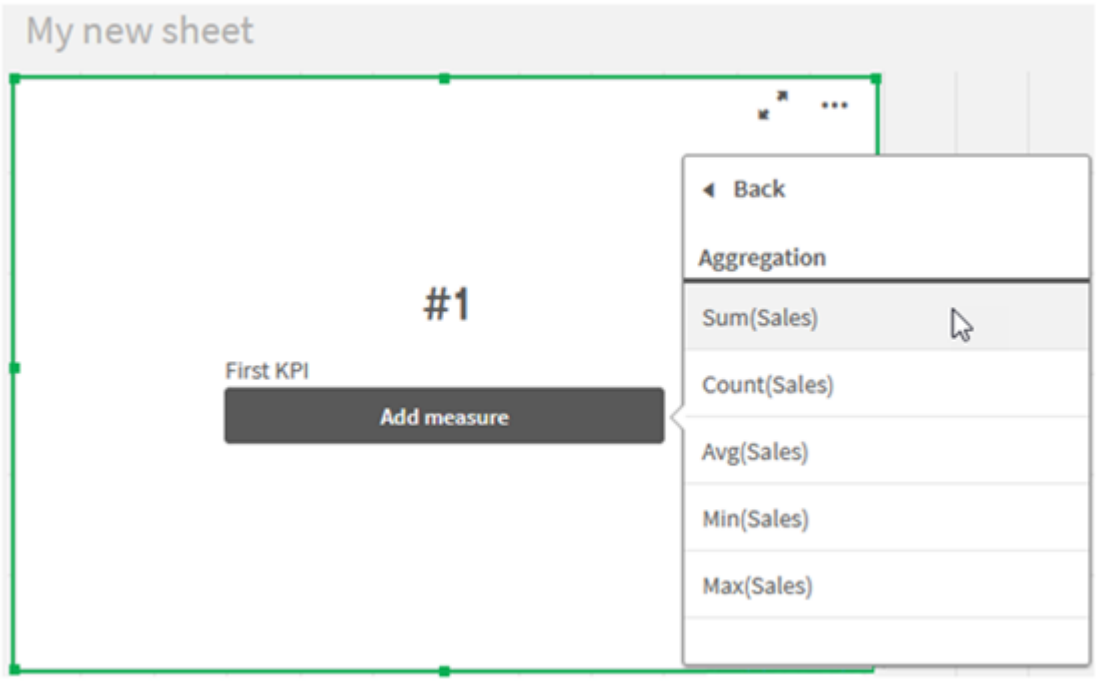
1. 시트의 탐색 막대에서 **자산** 패널을 연 다음 **차트**를 클릭합니다.



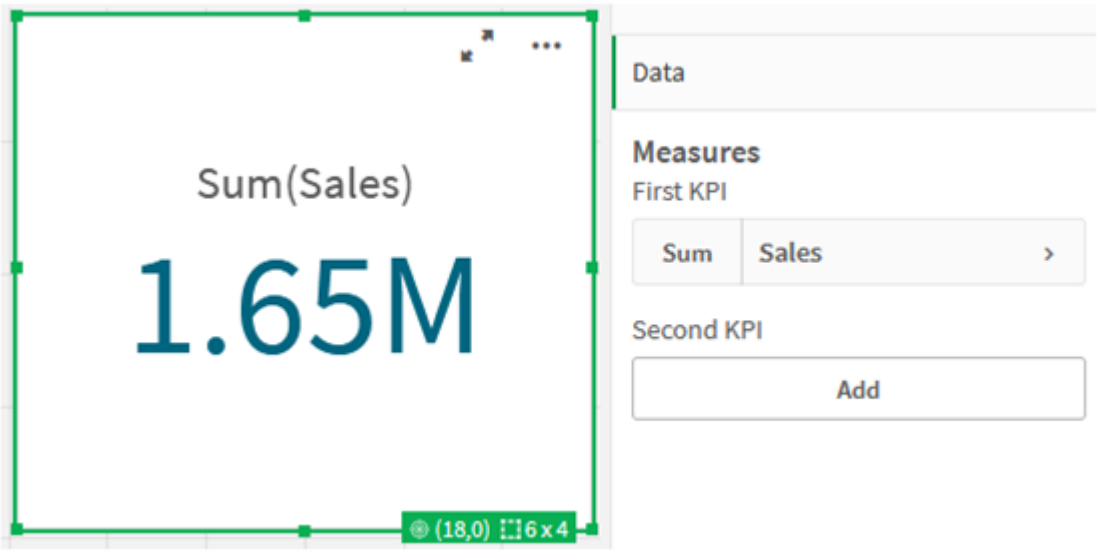
2. **KPI**를 시트로 끌어온 다음 **측정값 추가**를 클릭합니다.



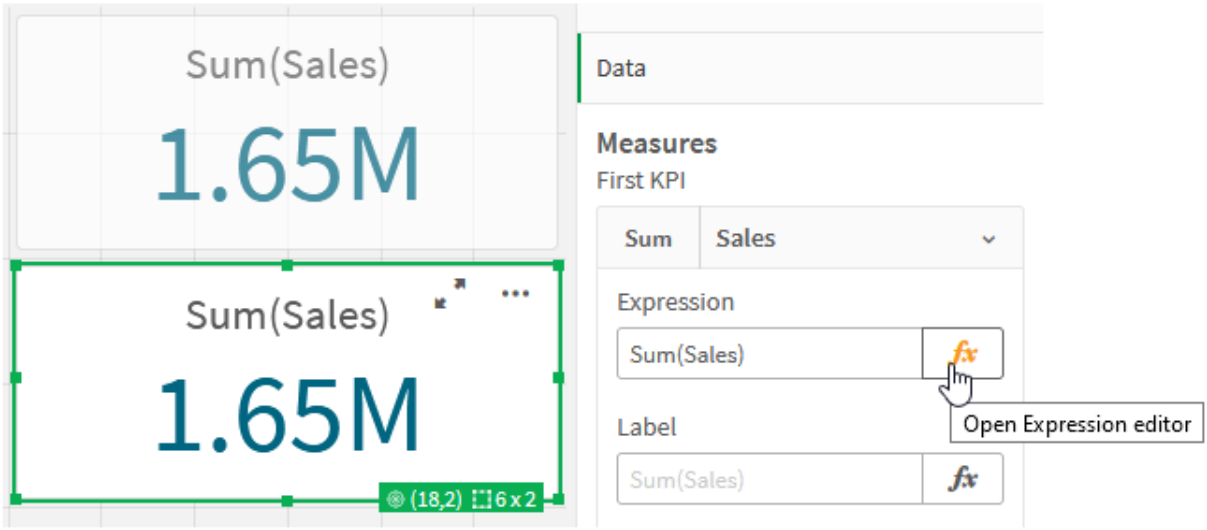
3. `Sales`를 클릭한 다음 집계에 대한 `Sum(Sales)`를 선택합니다.



KPI는 모든 연도의 판매량 합계를 보여 줍니다.



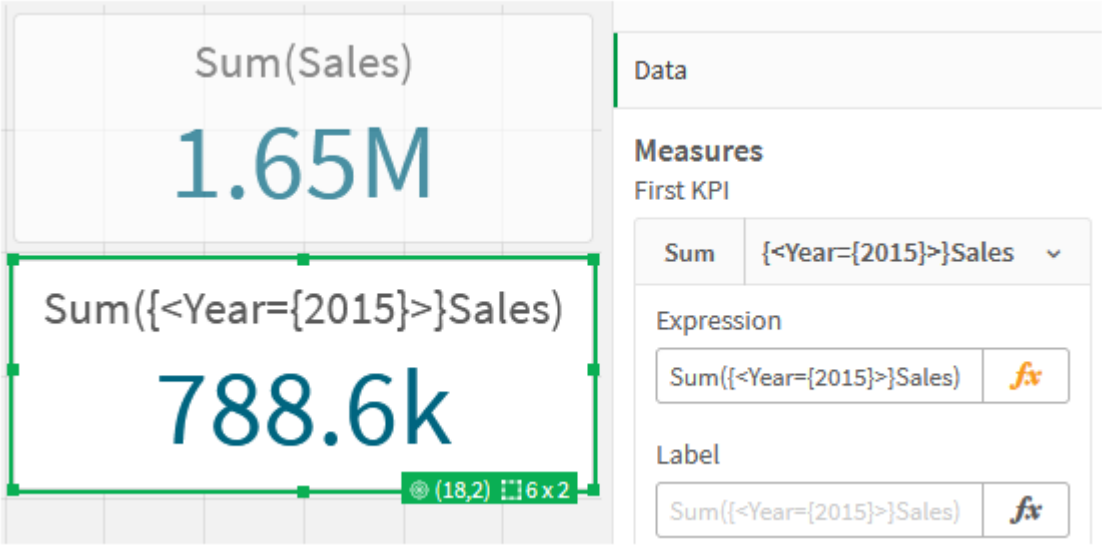
4. KPI를 복사 후 붙여넣어 새 KPI를 만듭니다.
5. 새 KPI를 클릭하고 **측정값**에서 **판매**를 클릭한 다음 **식 편집기 열기**를 클릭합니다.



식 편집기가 `Sum(Sales)` 집계와 함께 열립니다.

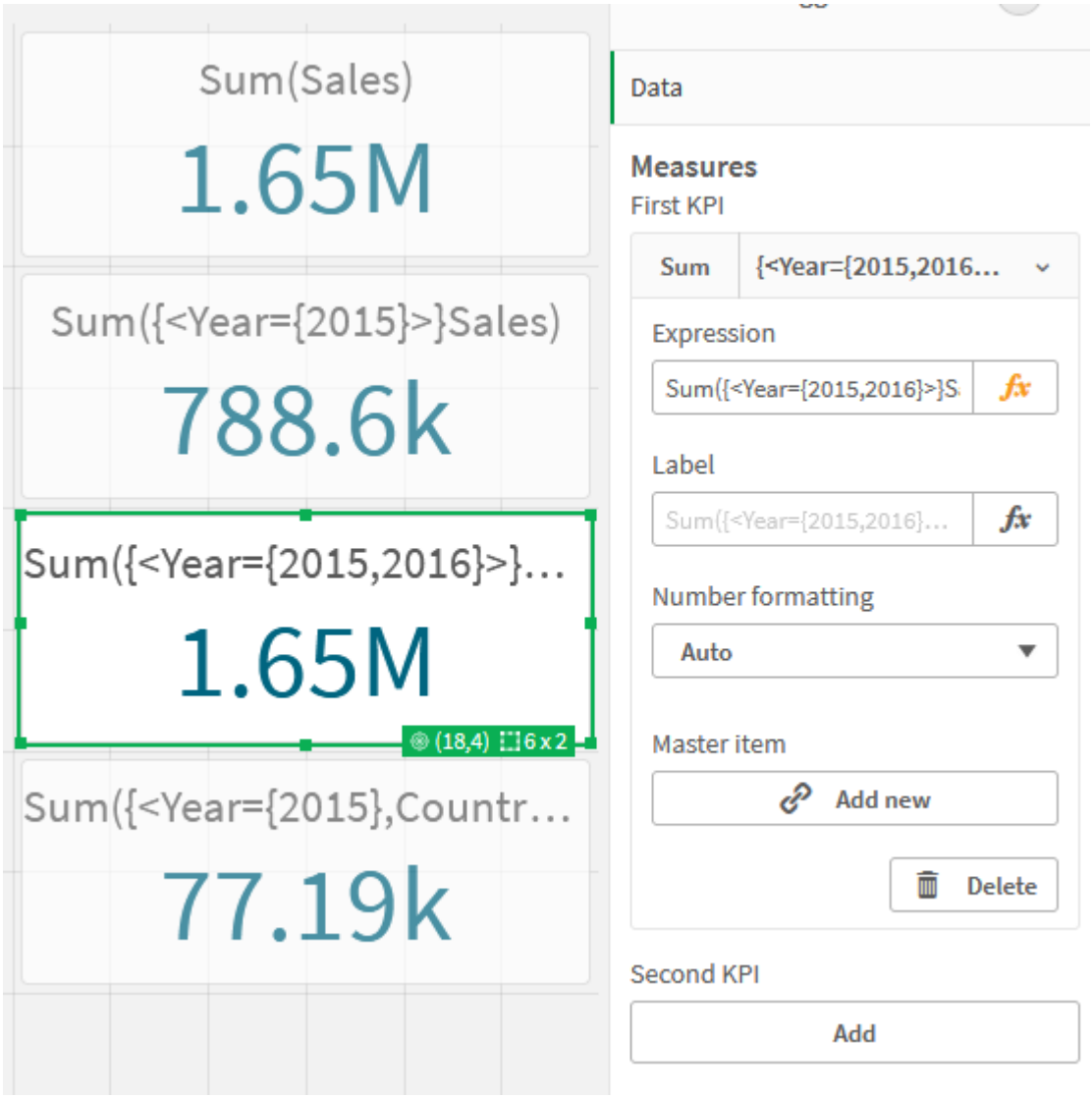
6. 식 편집기에서 2015년에 대해서만 Sales의 합계를 구하는 표현식을 만듭니다.
   i. 집합 표현식을 나타내는 중괄호를 추가합니다. `Sum({}Sales)`
   i. 집합 수정자를 나타내기 위해 꺾쇠 괄호를 추가합니다. `Sum({<>}Sales)`
   ii. 꺾쇠 괄호 안에 선택할 필드를 추가합니다. 이 경우 필드는 `Year`이고 그 뒤에 등호가 옵니다. 다음으로 2015를 다른 중괄호 세트로 묶습니다. 결과 집합 수정자는 `{<Year={2015}>}`입니다. 전체 표현식은 다음과 같습니다.
   `Sum({<Year={2015}>}Sales)`

iii. **적용**을 클릭하여 표현식을 저장하고 식 편집기를 닫습니다. 2015년의 Sales 합계는 KPI에 나와 있습니다.



7. 다음 표현식을 사용하여 두 개의 KPI를 더 만듭니다.
   `Sum({<Year={2015,2016}>}Sales)`
   위의 수정자는 `<Year={2015,2016}>`입니다. 표현식은 2015년과 2016년에 대한 Sales의 합계를 반환합니다.
   `Sum({<Year={2015},Country={'Germany'}>} Sales)`
   위의 수정자는 `<Year={2015}, Country={'Germany'}>`입니다. 표현식은 2015년이 Germany와 교차하는 2015년의 Sales 합계를 반환합니다.

집합 수정자를 사용한 KPI



집합 식별자 추가

식별자가 사용되지 않았기 때문에 위의 집합 표현식은 현재 선택을 기본으로 사용합니다. 다음으로 선택 시 동작을 지정하는 식별자를 추가합니다.

다음과 같이 하십시오.

시트에서 다음 집합 표현식을 작성하거나 복사합니다.

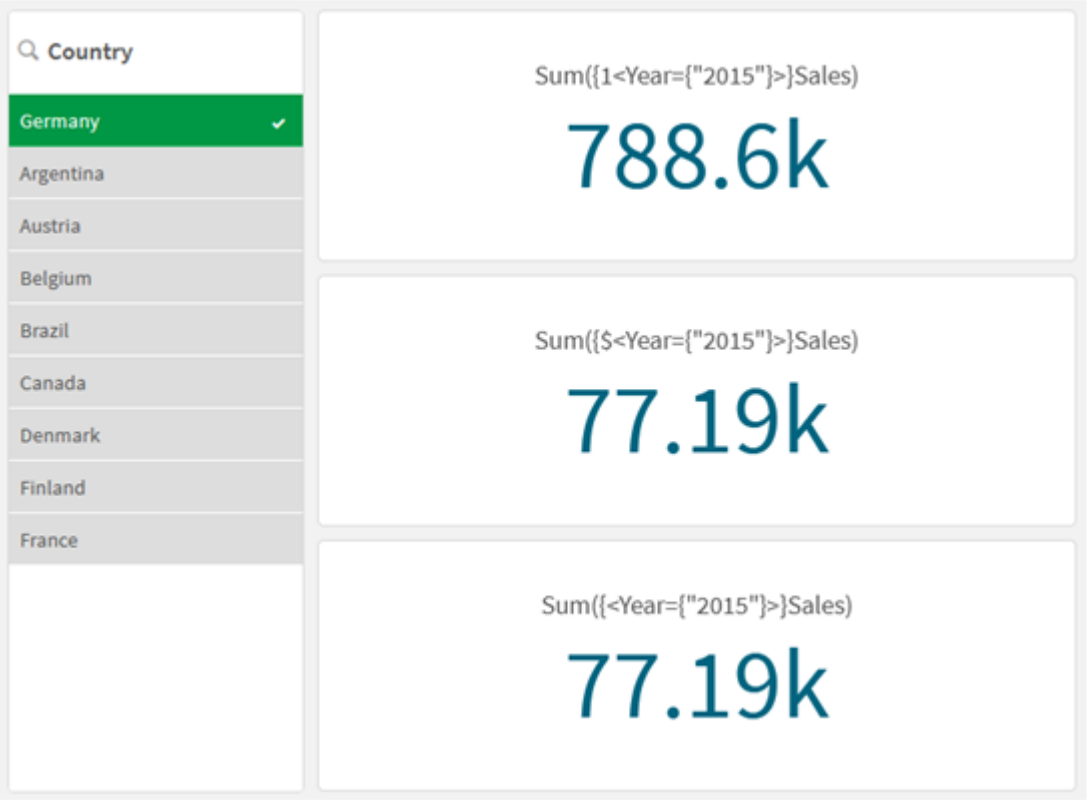
```
Sum({$<Year={"2015"}>}Sales)
```

`$` 식별자는 데이터에서 만들어진 현재 선택을 집합 표현식의 기반으로 설정합니다. 이는 식별자가 사용되지 않을 경우 기본 동작이기도 합니다.

```
Sum({1<Year={"2015"}>}Sales)
```

1 식별자는 2015년에 Sum(Sales)의 집계가 현재 선택된 내용을 무시하도록 합니다. 사용자가 다른 항목을 선택하더라도 집계 값은 변경되지 않습니다. 예를 들어 아래에서 Germany를 선택하면 2015년의 집계 합계 값이 변경되지 않습니다.

집합 수정자 및 식별자를 사용하는 KPI



연산자 추가

집합 연산자는 데이터 집합을 포함, 제외 또는 교차하는 데 사용됩니다. 모든 연산자는 집합을 피연산자로 사용하며 집합을 결과로 반환합니다.

두 가지 다른 상황에서 집합 연산자를 사용할 수 있습니다.

- 데이터의 레코드 집합을 나타내는 집합 식별자에 대한 집합 작업을 수행합니다.
- 요소 집합, 필드 값 또는 집합 수정자 내부에서 집합 작업을 수행합니다.

다음과 같이 하십시오.

시트에서 다음 집합 표현식을 작성하거나 복사합니다.

```
Sum({$<Year={2015}>+1<Country={'Germany'}>}Sales)
```

더하기 기호 (+) 연산자는 2015 및 Germany에 대한 데이터 집합의 합집합을 생성합니다. 위의 집합 식별자로 설명했듯이 달러 기호 ($) 식별자는 현재 선택이 첫 번째 피연산자 <Year={2015}>에 사용됨을 의미합니다. 1 식별자는 두 번째 피연산자 <Country={'Germany'}>에 대한 선택이 무시됨을 의미합니다.

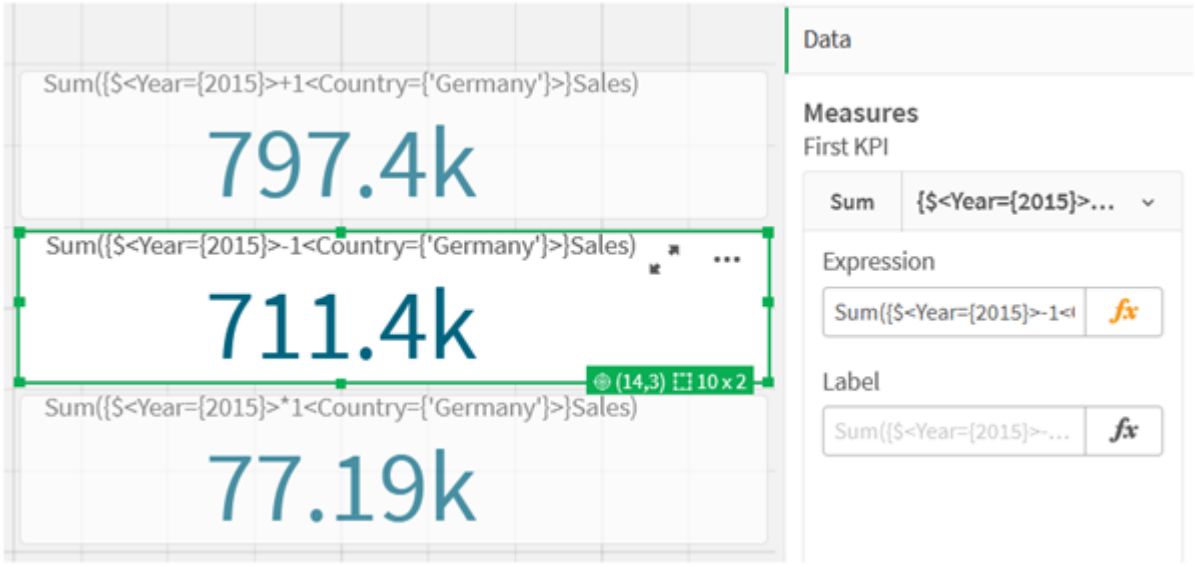
더하기 기호(+) 연산자를 사용하는 KPI

또는 빼기 기호 (-)를 사용하여 2015에 속하지만 Germany에 속하지 않는 레코드로 구성된 데이터 집합을 반환합니다. 또는 별표 (*)를 사용하여 두 집합에 모두 속하는 레코드로 구성된 집합을 반환합니다.

```
Sum({$<Year={2015}>-1<Country={'Germany'}>}Sales)
```

```
Sum({$<Year={2015}>*1<Country={'Germany'}>}Sales)
```

연산자를 사용하는 KPI



집합 표현식 자습서 데이터

로드 스크립트

다음 데이터를 인라인 로드로 로드한 다음 자습서에서 차트 표현식을 만듭니다.

```
//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
];
```

집합 표현식의 구문

전체 구문(우선 순위를 정의하기 위해 표준 괄호를 옵션으로 사용하는 경우 제외)은 Backus-Naur 이론을 사용하여 설명됩니다.

```
set_expression ::= { set_entity { set_operator set_entity } }
set_entity ::= set_identifier [ set_modifier ] | set_modifier
set_identifier ::= 1 | $ | $N | $_N | bookmark_id | bookmark_name
set_operator ::= + | - | * | /
set_modifier ::= < field_selection {, field_selection } >
field_selection ::= field_name [ = | += | -= | *= | /= ] element_set_
expression
element_set_expression ::= [ - ] element_set { set_operator element_set }
element_set ::= [ field_name ] | { element_list } | element_function
element_list ::= element { , element }
element_function ::= ( P | E ) ( [set_expression] [field_name] )
element ::= field_value | " search_mask "
```

6.3 차트 표현식의 일반적인 구문

다음 일반 구문 구조를 여러 선택적 매개 변수와 함께 차트 표현식에 사용할 수 있습니다.

```
expression ::= ( constant | expressionname | operator1 expression | expression operator2
expression | function | aggregation function | (expression ) )
```
설명:

constant는 곧은 작은따옴표로 묶인 문자열(텍스트, 날짜 또는 시간)이나 숫자입니다. 상수에는 천 단위 구분 기호가 사용되지 않으며,소수점 구분 기호인 소수점이 사용됩니다.

expressionname은 동일한 차트 내에 있는 다른 표현식의 이름(레이블)입니다.

operator1은 단항 연산자입니다(오른쪽에 있는 하나의 표현식에서 작동).

operator2는 이항 연산자입니다(양쪽에 하나씩 있는 두 개의 표현식에서 작동).

```
function ::= functionname ( parameters )
parameters ::= expression { , expression }
```
파라메타의 수와 유형은 임의가 아니라, 사용되는 함수에 따라 결정됩니다.

```
aggregationfunction ::= aggregationfunctionname ( parameters2 )
parameters2 ::= aggrexpression { , aggrexpression }
```
파라메타의 수와 유형은 임의가 아니라, 사용되는 함수에 따라 결정됩니다.

6.4 집계의 일반적인 구문

다음 일반 구문 구조를 여러 선택적 매개 변수와 함께 집계에 사용할 수 있습니다.

```
aggrexpression ::= ( fieldref | operator1 aggrexpression | aggrexpression operator2
aggrexpression | functioninaggr | ( aggrexpression ) )
```
fieldref는 필드 이름입니다.

```
functionaggr ::= functionname ( parameters2 )
```
fieldref가 항상 정확히 하나의 집계 함수로 둘러싸여 있고, 표현식이 해석 가능한 값을 반환하며, Qlik Sense에서 오류 메시지를 표시하지 않는다는 전제 하에 표현식과 함수는 자유롭게 중첩될 수 있습니다.

7 연산자

이 섹션에서는 Qlik Sense에서 사용할 수 있는 연산자에 대해 설명합니다. 연산자는 다음 두 가지 유형이 있습니다.

- 단항 연산자(피연산자 1개)
- 이항 연산자(피연산자 2개)

대부분의 연산자는 이항 연산자입니다.

다음과 같은 연산자를 정의할 수 있습니다.

- 비트 연산자
- 논리 연산자
- 숫자 연산자
- 관계형 연산자
- 문자열 연산자

7.1 비트 연산자

모든 비트 연산자는 피연산자를 부호 있는 정수(32비트)로 변환(절사)하고 동일한 방법으로 결과를 반환합니다. 모든 연산은 비트 단위로 수행됩니다. 피연산자를 숫자로 해석할 수 없는 경우 해당 연산은 NULL을 반환합니다.

비트 연산자

연산자	전체 이름	설명
bitnot	비트 부정.	단항 연산자입니다. 이 연산은 비트 단위로 수행한 피연산자의 논리 부정을 반환합니다. `bitnot 17`은 -18을 반환합니다.
bitand	비트 논리 곱.	이 연산은 비트 단위로 수행한 피연산자의 논리 곱을 반환합니다. `17 bitand 7`은 1을 반환합니다.
bitor	비트 논리 합.	이 연산은 비트 단위로 수행한 피연산자의 논리 합을 반환합니다. `17 bitor 7`은 23을 반환합니다.

연산자	전체 이름	설명
bitxor	비트 배타적 논리 합.	이 연산은 비트 단위로 수행한 피연산자의 배타적 논리 합을 반환합니다. `17 bitxor 7`은 22를 반환합니다.
>>	비트 오른쪽 시프트.	오른쪽으로 시프트된 첫 번째 피연산자를 반환합니다. 단계 수는 두 번째 피연산자로 정의합니다. `8 >> 2`는 2를 반환합니다.
<<	비트 왼쪽 시프트.	왼쪽으로 시프트된 첫 번째 피연산자를 반환합니다. 단계 수는 두 번째 피연산자로 정의합니다. `8 << 2`는 32를 반환합니다.

7.2 논리 연산자

모든 논리 연산자는 논리적으로 피연산자를 해석하며 True(-1) 또는 False(0)를 결과로 반환합니다.

논리 연산자

연산자	설명
not	논리 부정. 단항 연산자 중 하나입니다. 이 연산은 피연산자의 논리 부정을 반환합니다.
and	논리 곱. 이 연산은 피연산자의 논리 곱을 반환합니다.
or	논리 합. 이 연산은 피연산자의 논리 합을 반환합니다.
Xor	배타적 논리 합. 이 연산은 피연산자의 배타적 논리 합을 반환합니다. 즉, 논리 합과 유사하지만 피연산자가 모두 True일 경우 결과는 False라는 점이 다릅니다.

7.3 숫자 연산자

모든 숫자 연산자는 피연산자의 숫자 값을 사용하고 결과 값으로 숫자 값을 반환합니다.

숫자 연산자

연산자	설명
+	양수(단항 연산자) 또는 산술 더하기를 나타내는 기호. 이항 연산 시 피연산자 두 수의 합을 반환합니다.
-	음수(단항 연산자) 또는 산술 빼기를 나타내는 기호. 단항 연산 시 피연산자에 -1을 곱한 값을 반환하고, 이항 연산 시 피연산자 두 수 간의 차이 값을 반환합니다.
*	산술 곱셈 기호. 연산 시 피연산자 두 수의 곱셈 결과를 반환합니다.
/	산술 나눗셈 기호. 연산 시 피연산자 두 수 간의 비율 값을 반환합니다.

7.4 관계형 연산자

모든 관계형 연산자는 피연산자 값을 비교하여 그 결과로True(-1) 또는 False(0)를 반환합니다. 모든 관계형 연산자는 이항 연산자입니다.

관계형 연산자

연산자	설명
<	보다 작음 두 피연산자 모두 수치적으로 해석할 수 있으면 숫자 비교를 실행합니다. 이 연산은 비교 평가에 대한 논리 값을 반환합니다.
<=	보다 작거나 같음 두 피연산자 모두 수치적으로 해석할 수 있으면 숫자 비교를 실행합니다. 이 연산은 비교 평가에 대한 논리 값을 반환합니다.
>	보다 큼 두 피연산자 모두 수치적으로 해석할 수 있으면 숫자 비교를 실행합니다. 이 연산은 비교 평가에 대한 논리 값을 반환합니다.
>=	보다 크거나 같음 두 피연산자 모두 수치적으로 해석할 수 있으면 숫자 비교를 실행합니다. 이 연산은 비교 평가에 대한 논리 값을 반환합니다.
=	같음 두 피연산자 모두 수치적으로 해석할 수 있으면 숫자 비교를 실행합니다. 이 연산은 비교 평가에 대한 논리 값을 반환합니다.
<>	동일하지 않음 두 피연산자 모두 수치적으로 해석할 수 있으면 숫자 비교를 실행합니다. 이 연산은 비교 평가에 대한 논리 값을 반환합니다.

연산자	설명
precedes	**<** 연산자와는 달리 비교 연산 전에 인수 값의 수치적 해석을 시도하지 않습니다. 연산의 결과로 연산자의 왼쪽 값이 텍스트로 표시되고, 문자열 비교 시 오른쪽 값의 텍스트 표시보다 앞에 오면 true를 반환합니다. '1 ' precedes ' 2' 은 FALSE를 반환합니다. ' 1' precedes ' 2'은 TRUE를 반환합니다. 공백(' ')의 ASCII 값이 숫자의 ASCII 값보다 작은 값이기 때문입니다. 다음과 비교해 보십시오. '1 ' < ' 2' 는 TRUE를 반환합니다. ' 1' < ' 2'은 TRUE를 반환합니다.
follows	**>** 연산자와는 달리 비교 연산 전에 인수 값의 수치적 해석을 시도하지 않습니다. 연산의 결과로 연산자의 왼쪽 값이 텍스트로 표시되고, 문자열 비교 시 오른쪽 값의 텍스트 표시보다 뒤에 오면 true를 반환합니다. ' 2' follows '1'은 FALSE를 반환합니다. ' 2' follows ' 1'은 TRUE를 반환합니다. 공백(' ')의 ASCII 값이 숫자의 ASCII 값보다 작은 값이기 때문입니다. 다음과 비교해 보십시오. ' 2' > ' 1'은 TRUE를 반환합니다. ' 2' > '1 '은 TRUE를 반환합니다.

7.5 문자열 연산자

두 가지 문자열 연산자를 사용할 수 있습니다. 하나는 피연산자의 문자열 값을 사용하고 결과 값으로 문자열을 반환하는 연산자입니다. 다른 하나는 피연산자를 비교하고 일치 정도를 표시하는 부울 값을 반환하는 연산자입니다.

&

문자열 연결. 이 연산은 두 개의 피연산자 문자열이 나란히 연결된 텍스트 문자열을 반환합니다.

'abc' & 'xyz'는 'abcxyz'를 반환합니다.

like

와일드카드 문자를 사용하여 문자열 비교. 이 연산은 연산자 앞의 문자열이 연산자 뒤의 문자열과 일치하면 부울 True(-1)를 반환합니다. 두 번째 문자열은 와일드카드 문자 *(여러 개의 임의 문자) 또는 ?(하나의 임의 문자)를 포함할 수 있습니다.

'abc' like 'a*'는 True(-1)를 반환합니다.

'abcd' like 'a?c*'는 True(-1)를 반환합니다.

'abc' like 'a??bc'는 False(0)를 반환합니다.

8 스크립트 및 차트 함수

데이터 로드 스크립트 및 차트 표현식의 함수를 사용하여 데이터를 변환하고 집계합니다.

많은 함수를 데이터 로드 스크립트와 차트 표현식에서 모두 동일한 방법으로 사용할 수 있지만 여러 가지 예외가 있습니다.

- 일부 함수는 - 스크립트 함수로 지정된 데이터 로드 스크립트에서만 사용할 수 있습니다.
- 일부 함수는 - 차트 함수로 지정된 차트 표현식에서만 사용할 수 있습니다.
- 일부 함수는 데이터 로드 스크립트와 차트 표현식에서 모두 사용할 수 있지만 파라메타와 응용 프로그램에서 차이가 있습니다. 이러한 함수에 대해서는 - 스크립트 함수 또는 - 차트 함수로 지정된 별도의 항목에서 설명합니다.

8.1 서버 측 확장(SSE)을 위한 분석 연결

분석 연결에 의해 활성화된 함수는 분석 연결을 구성하고 Qlik Sense가 시작된 경우에만 표시됩니다.

QMC에서 분석 연결을 구성합니다. Qlik Sense 사이트 관리 가이드의 “분석 연결 만들기” 항목을 참조하십시오.

Qlik Sense Desktop에서 *Settings.ini* 파일을 편집하여 분석 연결을 구성합니다. Qlik Sense Desktop 가이드의 “Qlik Sense Desktop에서 분석 연결 구성” 항목을 참조하십시오.

8.2 집계 함수

집계 함수로 알려진 함수 제품군은 여러 필드 값을 입력으로 사용하고 그룹당 단일 결과를 반환하는 함수로 구성됩니다. 여기서 그룹화는 차트 차원 또는 스크립트 문에서 **group by** 절로 정의됩니다.

집계 함수에는 **Sum()**, **Count()**, **Min()**, **Max()** 등이 있습니다.

대부분의 집계 함수는 데이터 로드 스크립트와 차트 표현식에서 모두 사용할 수 있지만 구문이 다릅니다.

제한 사항:

집계 함수의 매개 변수는 이러한 내부 집계에 **TOTAL** 한정자가 포함되어 있지 않는 한 다른 집계 함수를 포함하지 않아야 합니다. 고급 중첩 집계가 필요한 경우는 고급 함수 **Aggr**을 지정된 차원과 함께 사용하십시오.

엔터티 이름을 지정할 때 둘 이상의 필드, 변수 또는 측정값에 동일한 이름을 지정하지 마십시오. 동일한 이름을 가진 엔터티 간의 충돌을 해결하기 위한 엄격한 우선 순위가 있습니다. 이 순서는 이러한 엔터티가 사용되는 모든 개체 또는 컨텍스트에 반영됩니다. 이 우선 순위는 다음과 같습니다.

- 집계 내에서 필드는 변수보다 우선합니다. 측정값 레이블은 집계와 관련이 없으며 우선 순위가 지정되지 않습니다.
- 집계 외부에서 측정값 레이블은 변수보다 우선 순위가 높으며, 이 변수는 필드 이름보다 우선합니다.

- 또한 집계 외부에서 레이블이 실제로 계산된 것이 아닌 한 해당 레이블을 참조하여 측정값을 재사용할 수 있습니다. 이 경우 자체 참조의 위험을 줄이기 위해 측정값의 중요성이 떨어지며, 이 경우 이름은 항상 첫 번째는 측정값 레이블, 두 번째는 필드 이름, 세 번째는 변수 이름으로 해석됩니다.

데이터 로드 스크립트에서 집계 함수 사용

집계 함수는 **LOAD** 및 **SELECT** 문 내에서만 사용할 수 있습니다.

차트 표현식에서 집계 함수 사용

집계 함수의 매개 변수는 이러한 내부 집계에 **TOTAL** 한정자가 포함되어 있지 않는 한 다른 집계 함수를 포함하지 않아야 합니다. 고급 중첩 집계가 필요한 경우는 고급 함수 **Aggr**을 지정된 차원과 함께 사용하십시오.

집계 함수는 선택에 의해 정의된 사용 가능한 레코드의 집합을 집계합니다. 하지만 집합 분석 시 집합 표현식을 사용하여 대체 레코드 집합을 정의할 수 있습니다.

집계 계산 방법

집계는 특정 테이블의 레코드를 반복하여 그 안의 레코드를 집계합니다. 예를 들어 **Count**(<Field>)는 <Field>가 있는 테이블의 레코드 수를 계수합니다. 고유 필드 값만 집계하려는 경우 **distinct** 절을 사용해야 합니다(예: **Count**(**distinct** <Field>)).

집계 함수에 다른 테이블의 필드가 포함된 경우 집계 함수는 필드로 구성된 테이블의 교차곱 레코드를 반복합니다. 이러한 집계 방식을 사용하면 성능이 저하되므로, 특히 대량의 데이터가 있는 경우에는 이러한 집계를 사용하지 않는 것이 좋습니다.

키 필드 집계

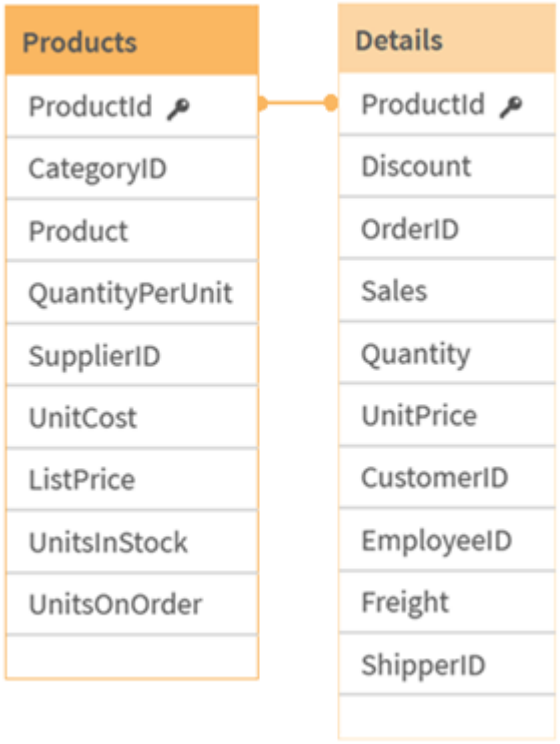
집계가 계산되는 방식은 집계에 사용해야 하는 테이블이 명확하지 않아 키 필드를 집계할 수 없음을 의미합니다. 예를 들어 <Key> 필드가 두 테이블을 연결하는 경우 **Count**(<Key>)는 첫 번째 테이블의 레코드 수를 반환해야 할지 두 번째 테이블의 레코드 수를 반환해야 할지 명확하지 않습니다.

하지만 **distinct** 절을 사용하면 집계는 잘 정의되고 계산할 수 있습니다.

따라서 **distinct** 절 없이 집계 함수 내에 키 필드를 사용하는 경우 Qlik Sense가 반환하는 숫자는 의미가 없을 수 있습니다. 해결 방법은 **distinct** 절을 사용하거나 하나의 테이블에만 있는 키 사본을 사용하는 것입니다.

예를 들어 다음 표에서는 ProductID는 테이블 간의 키입니다.

제품 및 세부 정보 테이블 간의 ProductID 키



Count(ProductID)는 Products 테이블(제품당 하나의 레코드만 있음 – ProductID가 기본 키임)에서 계산되거나 Details 테이블(제품당 여러 레코드가 있을 가능성이 가장 높음)에서 계산될 수 있습니다. 고유한 제품의 수를 계산하려면 Count(distinct ProductID)를 사용해야 합니다. 특정 테이블의 행 수를 계산하려면 키를 사용하지 않아야 합니다.

기본 집계 함수

기본 집계 함수 개요

기본 집계 함수는 가장 일반적인 집계 함수의 그룹입니다.

각 함수는 개요가 끝난 후에 더 자세히 설명합니다. 구문에서 함수 이름을 클릭하여 해당 함수에 대한 상세 설명에 즉시 액세스할 수도 있습니다.

데이터 로드 스크립트의 기본 집계 함수

FirstSortedValue

FirstSortedValue()는 단위 가격이 가장 낮은 제품의 이름 등과 같이 **sort_weight** 인수의 정렬 결과에 해당하는 **value**에서 지정된 표현식으로부터의 값을 반환합니다. 정렬 순서의 n번째 값은 **rank**에서 지정할 수 있습니다. 둘 이상의 결과 값이 지정된 **rank**에 대해 동일한 **sort_weight**를 공유하는 경우, 해당 함수는 NULL을 반환합니다. 정렬된 값은 **group by** 절로 정의된 레코드 수에 대해 반복되거나, **group by** 절이 정의되지 않은 경우 전체 데이터 셋에 대해 집계됩니다.

```
FirstSortedValue ([ distinct ] expression, sort_weight [, rank ])
```

Max

Max()는 **group by** 절로 정의된 표현식에서 집계된 데이터 중 가장 높은 숫자 값을 찾습니다. **rank** n을 지정하면 n번째 높은 값을 찾을 수 있습니다.

```
Max ( expression[, rank])
```

Min

Min()은 **group by** 절로 정의된 표현식에서 집계된 데이터 중 가장 낮은 숫자 값을 반환합니다. **rank** n을 지정하면 n번째 낮은 값을 찾을 수 있습니다.

```
Min ( expression[, rank])
```

Mode

Mode()는 **group by** 절로 정의된 표현식에서 집계된 데이터 중 가장 발생 빈도가 높은 값, 즉 모드 값을 반환합니다. **Mode()** 함수는 텍스트 값뿐 아니라 숫자 값도 반환할 수 있습니다.

```
Mode (expression )
```

Only

Only()는 집계된 데이터에서 사용 가능한 결과가 하나만 있는 경우 값을 반환합니다. 레코드에 하나의 값만 포함된 경우 해당 값이 반환되고, 그렇지 않으면 NULL이 반환됩니다. 여러 레코드에 대해 평가하려면 **group by** 절을 사용하십시오. **Only()** 함수는 숫자 및 텍스트 값을 반환할 수 있습니다.

```
Only (expression )
```

Sum

Sum()은 **group by** 절로 정의된 표현식에서 집계된 값의 총 합계를 계산합니다.

```
Sum ([distinct]expression)
```

차트 표현식의 기본 집계 함수

차트 집계 함수는 차트 표현식 내의 필드에서만 사용할 수 있습니다. 한 집계 함수의 인수 표현식에 다른 집계 함수가 포함되지 않아야 합니다.

FirstSortedValue

FirstSortedValue()는 단위 가격이 가장 낮은 제품의 이름 등과 같이 **sort_weight** 인수의 정렬 결과에 해당하는 **value**에서 지정된 표현식으로부터의 값을 반환합니다. 정렬 순서의 n번째 값은 **rank**에서 지정할 수 있습니다. 둘 이상의 결과 값이 지정된 **rank**에 대해 동일한 **sort_weight**를 공유하는 경우, 해당 함수는 NULL을 반환합니다.

```
FirstSortedValue - 차트 함수([{SetExpression}] [DISTINCT] [TOTAL [<fld
{,fld}>]] value, sort_weight [,rank])
```

Max

Max()는 집계된 데이터의 가장 높은 값을 찾습니다. **rank** n을 지정하면 n번째 높은 값을 찾을 수 있습니다.

```
Max - 차트 함수Max()는 집계된 데이터의 가장 높은 값을 찾습니다. rank n을 지정하면 n번째
높은 값을 찾을 수 있습니다. Max 함수와 유사한 기능을 가진 FirstSortedValue 및
rangemax도 검토하는 것이 좋습니다. Max([{SetExpression}] [TOTAL [<fld {,fld}>]]
expr [,rank]) 숫자 인수인수설명expr측정할 데이터가 포함된 표현식 또는 필드입니
다.rankrank의 기본값은 1이며, 이는 가장 높은 값에 해당합니다. rank를 2로 지정하면 두 번
째로 높은 값이 반환됩니다. rank가 3이면 세 번째로 높은 값이 반환되며, 이런 식으로 이어집
```

니다.SetExpression기본적으로 집계 함수는 선택에 의해 정의된 사용 가능한 레코드의 집합을 집계합니다. 집합 분석 표현식으로 대체 레코드 집합을 정의할 수 있습니다. TOTALTOTAL이 함수 인수 앞에 오는 경우, 현재 선택을 고려하되 현재 차원 값에 관련되지 않은 가능한 모든 값에 대한 계산이 실행됩니다. 즉, 차트 차원은 무시됩니다. TOTAL [<fld {.fld}>](여기서 TOTAL 한정자 뒤에는 하나 이상의 필드 이름 목록이 차트 차원 변수의 하위 집합으로 옴)을 사용하여 가능한 전체 값의 하위 집합을 만듭니다. 데이터CustomerProductUnitSalesUnitPrice AstridaAA416AstridaAA1015AstridaBB99BetacabBB510BetacabCC220BetacabDD-25CanutilityAA815CanutilityCC-19예 및 결과예결과Max(UnitSales)10이며, UnitSales에서 가장 높은 값이기 때문입니다.순서의 값은 (UnitSales)에서 판매된 단품 수에 단품 가격을 곱한 수로 계산됩니다.Max(UnitSales*UnitPrice)150이며, (UnitSales)*(UnitPrice)의 가능한 모든 값을 계산한 결과 중에서 가장 높은 값이기 때문입니다.Max(UnitSales, 2)9이며, 두 번째로 높은 값이기 때문입니다.Max(TOTAL UnitSales)10이며, TOTAL 한정자는 차트 차원을 무시하고 가능한 가장 높은 값이 발견되었음을 의미하기 때문입니다. Customer를 차원으로 사용하는 차트의 경우, TOTAL 한정자를 통해 각 고객의 최대 UnitSales가 아니라 전체 데이터 집합에 걸친 최댓값이 반환되도록 할 수 있습니다. Customer B를 선택합니다.Max({1} TOTAL UnitSales)10이며 Set Analysis 표현식 {1}이 선택 내용과 관계없이 평가할 레코드의 집합을 ALL로 정의하므로 선택 내용과 무관합니다.예에서 사용된 데이터: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()은 집계된 데이터의 가장 낮은 값을 찾습니다. **rank** n을 지정하면 n번째 낮은 값을 찾을 수 있습니다.

Min - 차트 함수([{SetExpression}] [DISTINCT] [TOTAL [<fld {,fld}>]] expr [,rank]**)**

Mode

Mode()는 집계된 데이터에서 가장 발생 빈도가 높은 값 즉, 모드 값을 찾습니다. **Mode()** 함수는 숫자 값뿐 아니라 텍스트 값도 처리할 수 있습니다.

Mode - 차트 함수 ({[SetExpression] [**TOTAL** [<fld {,fld}>]]} expr**)**

Only

Only()는 집계된 데이터에서 사용 가능한 결과가 하나만 있는 경우 값을 반환합니다. 예를 들어, 단위 가격=9인 제품만 검색하는 경우 단위 가격이 9인 제품이 둘 이상 있으면 NULL을 반환합니다.

Only - 차트 함수([{SetExpression}] [DISTINCT] [TOTAL [<fld {,fld}>]] expr**)**

Sum

Sum()은 집계 데이터 전체에서 필드 또는 표현식을 통해 지정된 값의 합계를 계산합니다.

Sum - 차트 함수([{SetExpression}] [DISTINCT] [TOTAL [<fld {,fld}>]] expr]**)**

FirstSortedValue

FirstSortedValue()는 단위 가격이 가장 낮은 제품의 이름 등과 같이 **sort_weight** 인수의 정렬 결과에 해당하는 **value**에서 지정된 표현식으로부터의 값을 반환합니다. 정렬 순서의 n번째 값은 **rank**에서 지정할 수 있습니다. 둘 이상의 결과 값이 지정된 **rank**에 대해 동일한 **sort_weight**를 공유하는 경우, 해당 함수는 NULL을 반환합니다. 정렬된 값은 **group by** 절로 정의된 레코드 수에 대해 반복되거나, **group by** 절이 정의되지 않은 경우 전체 데이터 셋에 대해 집계됩니다.

구문:

```
FirstSortedValue ([ distinct ] value, sort-weight [, rank ])
```

반환 데이터 유형: dual

인수:

인수

인수	설명
value Expression	이 함수는 **sort_weight**를 정렬한 결과에 해당하는 **value** 표현식의 값을 찾습니다.
sort-weight Expression	저장할 데이터가 포함된 표현식입니다. 해당하는 **value** 표현식 값을 결정할 **sort_weight**의 첫 번째(가장 낮은) 값을 찾습니다. **sort_weight** 앞에 빼기 기호가 있는 경우, 이 함수는 마지막(가장 높은) 정렬 값을 반환합니다.
rank Expression	**rank** "n"을 1보다 큰 수로 지정하면 정렬된 n번째 값을 얻을 수 있습니다.
distinct	함수 인수 앞에 **DISTINCT**라는 단어가 있을 경우 해당 함수 인수의 평가 결과로 생성된 중복이 무시됩니다.

예 및 결과:

예제 스크립트를 앱에 추가하고 실행합니다. 그런 다음, 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.

아래의 결과 열과 동일한 결과를 얻으려면 속성 패널의 정렬 아래에서 자동을 사용자 지정으로 전환한 후 숫자순 및 사전순 정렬을 선택 취소합니다.

스크립팅 예

예	결과
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 이 함수는 UnitSales를 가장 작은 값부터 큰 값까지 순서대로 정렬하여, 가장 작은 UnitSales 값(최소 주문)을 가진 Customer 값을 찾습니다. CC가 고객 Astrida의 최소 주문(UnitSales 값=2)에 해당하기 때문입니다. AA는 고객 Betacab의 최소 주문(4)에 해당하고, AA는 고객 Canutility의 최소 주문(8)에 해당하고, DD는 고객 Divadip.의 최소 주문(10)에 해당합니다.
이전 예에서처럼 **Temp** 테이블이 로드된 것으로 가정합니다. LOAD Customer,FirstSortedValue(Product, -UnitSales) as MyProductWithLargestOrderByCustomer Resident Temp Group By Customer;	Customer MyProductWithLargestOrderByCustomer Astrida AA Betacab DD Canutility CC Divadip - 빼기 기호가 sort_weight 인수 앞에 오므로 이 함수는 가장 큰 값부터 정렬합니다. AA는 고객 Astrida의 최대 주문(UnitSales 값: 18)에 해당하고, DD는 고객 Betacab의 최대 주문(12)에 해당하고, CC는 고객 Canutility의 최대 주문(13)에 해당하기 때문입니다. 고객 Divadip의 최대 주문(16)에 대해 동일한 값이 2개 있으므로 결과는 Null입니다.
이전 예에서처럼 **Temp** 테이블이 로드된 것으로 가정합니다. LOAD Customer,FirstSortedValue(distinct Product, -UnitSales) as MyProductWithSmallestOrderByCustomer Resident Temp Group By Customer;	Customer MyProductWithLargestOrderByCustomer Astrida AA Betacab DD Canutility CC Divadip AA distinct 한정자를 사용한 것을 빼면 이전 예와 동일합니다. 따라서 Divadip의 중복된 결과가 무시되고 Null 외의 값이 반환됩니다.

FirstSortedValue - 차트 함수

FirstSortedValue()는 단위 가격이 가장 낮은 제품의 이름 등과 같이 **sort_weight** 인수의 정렬 결과에 해당하는 **value**에서 지정된 표현식으로부터의 값을 반환합니다. 정렬 순서의 n번째 값은 **rank**에서 지정할 수 있습니다. 둘 이상의 결과 값이 지정된 **rank**에 대해 동일한 **sort_weight**를 공유하는 경우, 해당 함수는 NULL을 반환합니다.

구문:

```
FirstSortedValue([{SetExpression}] [DISTINCT] [TOTAL [<fld {,fld}>]] value,
sort_weight [,rank])
```

반환 데이터 유형: dual

인수:

인수

인수	설명
value	출력 필드 이 함수는 **sort_weight**를 정렬한 결과에 해당하는 **value** 표현식의 값을 찾습니다.
sort_weight	입력 필드 저장할 데이터가 포함된 표현식입니다. 해당하는 **value** 표현식 값을 결정할 **sort_weight**의 첫 번째(가장 낮은) 값을 찾습니다. **sort_weight** 앞에 빼기 기호가 있는 경우, 이 함수는 마지막(가장 높은) 정렬 값을 반환합니다.
rank	**rank** "n"을 1보다 큰 수로 지정하면 정렬된 n번째 값을 얻을 수 있습니다.
SetExpression	기본적으로 집계 함수는 선택에 의해 정의된 사용 가능한 레코드의 집합을 집계합니다. 집합 분석 표현식으로 대체 레코드 집합을 정의할 수 있습니다.
DISTINCT	함수 인수 앞에 **DISTINCT**라는 단어가 있을 경우 해당 함수 인수의 평가 결과로 생성된 중복이 무시됩니다.
TOTAL	**TOTAL**이 함수 인수 앞에 오는 경우, 현재 선택을 고려하되 현재 차원 값에 관련되지 않은 가능한 모든 값에 대한 계산이 실행됩니다. 즉, 차트 차원은 무시됩니다. **TOTAL [<fld {.fld}>]**(여기서 **TOTAL** 한정자 뒤에는 하나 이상의 필드 이름 목록이 차트 차원 변수의 하위 집합으로 옴)을 사용하여 가능한 전체 값의 하위 집합을 만듭니다.

예 및 결과:

데이터

Customer	Product	UnitSales	UnitPrice
Astrida	AA	4	16
Astrida	AA	10	15
Astrida	BB	9	9

Customer	Product	UnitSales	UnitPrice
Betacab	BB	5	10
Betacab	CC	2	20
Betacab	DD	-	25
Canutility	AA	8	15
Canutility	CC	-	19

예 및 결과

예	결과
`firstsortedvalue (Product, UnitPrice)`	BB이며, `UnitPrice`(9)가 가장 낮은 `Product`입니다.
`firstsortedvalue (Product, UnitPrice, 2)`	BB이며, `UnitPrice`(10)가 두 번째로 낮은 `Product`입니다.
`firstsortedvalue (Customer, -UnitPrice, 2)`	Betacab이며, `UnitPrice`(20)가 두 번째로 높은 `Product`를 보유한 `Customer`입니다.
`firstsortedvalue (Customer, UnitPrice, 3)`	NULL이며 rank(세 번째로 낮음) `UnitPrice`(15)가 동일한 `Customer` (Astrida 및 Canutility)의 값이 두 개이기 때문입니다. distinct 한정자를 사용하여 예상치 못한 null 결과가 발생하지 않도록 하십시오.
`firstsortedvalue (Customer, -UnitPrice*UnitSales, 2)`	Canutility이며, `UnitPrice`에 `UnitSales`(120)를 곱한 판매 순서 값이 두 번째로 높은 `Customer`입니다.

예에서 사용된 데이터:

```
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()는 **group by** 절로 정의된 표현식에서 집계된 데이터 중 가장 높은 숫자 값을 찾습니다. **rank** n을 지정하면 n번째 높은 값을 찾을 수 있습니다.

구문:

```
Max ( expr [, rank])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
expr Expression	측정할 데이터가 포함된 표현식 또는 필드입니다.
rank Expression	**rank**의 기본값은 1이며, 이는 가장 높은 값에 해당합니다. **rank**를 2로 지정하면 두 번째로 높은 값이 반환됩니다. **rank**가 3이면 세 번째로 높은 값이 반환되며, 이런 식으로 이어집니다.

예 및 결과:

예제 스크립트를 앱에 추가하고 실행합니다. 그런 다음, 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.

아래의 결과 열과 동일한 결과를 얻으려면 속성 패널의 정렬 아래에서 자동을 사용자 지정으로 전환한 후 숫자순 및 사전순 정렬을 선택 취소합니다.

```
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;
```

결과 테이블

Customer	MyMax
Astrida	18
Betacab	5
Canutility	8

이전 예에서처럼 **Temp** 테이블이 로드된 것으로 가정합니다.

```
LOAD Customer, Max(UnitSales,2) as MyMaxRank2 Resident Temp Group By Customer;
```

결과 테이블

Customer	**MyMaxRank2**
Astrida	10
Betacab	4
Canutility	-

Max - 차트 함수

Max()는 집계된 데이터의 가장 높은 값을 찾습니다. **rank** n을 지정하면 n번째 높은 값을 찾을 수 있습니다.

***Max** 함수와 유사한 기능을 가진 **FirstSortedValue** 및 **rangemax**도 검토하는 것이 좋습니다.*

구문:

```
Max([{SetExpression}] [TOTAL [<fld {,fld}>]] expr [,rank])
```

반환 데이터 유형: 숫자

인수:

인수

인수	**설명**
expr	측정할 데이터가 포함된 표현식 또는 필드입니다.
rank	**rank**의 기본값은 1이며, 이는 가장 높은 값에 해당합니다. **rank**를 2로 지정하면 두 번째로 높은 값이 반환됩니다. **rank**가 3이면 세 번째로 높은 값이 반환되며, 이런 식으로 이어집니다.
SetExpression	기본적으로 집계 함수는 선택에 의해 정의된 사용 가능한 레코드의 집합을 집계합니다. 집합 분석 표현식으로 대체 레코드 집합을 정의할 수 있습니다.
TOTAL	**TOTAL**이 함수 인수 앞에 오는 경우, 현재 선택을 고려하되 현재 차원 값에 관련되지 않은 가능한 모든 값에 대한 계산이 실행됩니다. 즉, 차트 차원은 무시됩니다. **TOTAL [<fld {.fld}>]**(여기서 **TOTAL** 한정자 뒤에는 하나 이상의 필드 이름 목록이 차트 차원 변수의 하위 집합으로 옴)을 사용하여 가능한 전체 값의 하위 집합을 만듭니다.

예 및 결과:

데이터

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

예 및 결과

예	결과
`Max(UnitSales)`	10이며, `UnitSales`에서 가장 높은 값이기 때문입니다.
순서의 값은 (`UnitSales`)에서 판매된 단품 수에 단품 가격을 곱한 수로 계산됩니다. `Max (UnitSales*UnitPrice)`	150이며, (`UnitSales`)*(`UnitPrice`)의 가능한 모든 값을 계산한 결과 중에서 가장 높은 값이기 때문입니다.
`Max(UnitSales, 2)`	9이며, 두 번째로 높은 값이기 때문입니다.
`Max(TOTAL UnitSales)`	10이며, TOTAL 한정자는 차트 차원을 무시하고 가능한 가장 높은 값이 발견되었음을 의미하기 때문입니다. Customer를 차원으로 사용하는 차트의 경우, TOTAL 한정자를 통해 각 고객의 최대 UnitSales가 아니라 전체 데이터 집합에 걸친 최댓값이 반환되도록 할 수 있습니다.
`Customer B`를 선택합니다. `Max({1} TOTAL UnitSales)`	10이며 Set Analysis 표현식 {1}이 선택 내용과 관계없이 평가할 레코드의 집합을 ALL로 정의하므로 선택 내용과 무관합니다.

예에서 사용된 데이터:

```
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 '|');
```

관련 항목:

- *FirstSortedValue - 차트 함수 (page 323)*
- *RangeMax (page 1295)*

Min

Min()은 **group by** 절로 정의된 표현식에서 집계된 데이터 중 가장 낮은 숫자 값을 반환합니다. **rank** n을 지정하면 n번째 낮은 값을 찾을 수 있습니다.

구문:

```
Min ( expr [, rank])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
expr Expression	측정할 데이터가 포함된 표현식 또는 필드입니다.
rank Expression	**rank**의 기본값은 1이며, 이는 가장 낮은 값에 해당합니다. **rank**를 2로 지정하면 두 번째로 낮은 값이 반환됩니다. **rank**가 3이면 세 번째로 낮은 값이 반환되며, 이런 식으로 이어집니다.

예 및 결과:

예제 스크립트를 앱에 추가하고 실행합니다. 그런 다음, 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.

아래의 결과 열과 동일한 결과를 얻으려면 속성 패널의 정렬 아래에서 자동을 사용자 지정으로 전환한 후 숫자순 및 사전순 정렬을 선택 취소합니다.

```
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;
```

결과 테이블

Customer	**MyMin**
Astrida	2
Betacab	4
Canutility	8

이전 예에서처럼 **Temp** 테이블이 로드된 것으로 가정합니다.

```
LOAD Customer, Min(UnitSales,2) as MyMinRank2 Resident Temp Group By Customer;
```

결과 테이블

Customer	**MyMinRank2**
Astrida	9
Betacab	5
Canutility	-

Min - 차트 함수

Min()은 집계된 데이터의 가장 낮은 값을 찾습니다. **rank** n을 지정하면 n번째 낮은 값을 찾을 수 있습니다.

***Min** 함수와 유사한 기능을 가진 **FirstSortedValue** 및 **rangemin**도 검토하는 것이 좋습니다.*

구문:

```
Min({[SetExpression] [TOTAL [<fld {,fld}>]]} expr [,rank])
```

반환 데이터 유형: 숫자

인수:

인수

인수	**설명**
expr	측정할 데이터가 포함된 표현식 또는 필드입니다.

인수	설명
rank	**rank**의 기본값은 1이며, 이는 가장 낮은 값에 해당합니다. **rank**를 2로 지정하면 두 번째로 낮은 값이 반환됩니다. **rank**가 3이면 세 번째로 낮은 값이 반환되며, 이런 식으로 이어집니다.
SetExpression	기본적으로 집계 함수는 선택에 의해 정의된 사용 가능한 레코드의 집합을 집계합니다. 집합 분석 표현식으로 대체 레코드 집합을 정의할 수 있습니다.
TOTAL	**TOTAL**이 함수 인수 앞에 오는 경우, 현재 선택을 고려하되 현재 차원 값에 관련되지 않은 가능한 모든 값에 대한 계산이 실행됩니다. 즉, 차트 차원은 무시됩니다. **TOTAL [<fld {.fld}>]**(여기서 **TOTAL** 한정자 뒤에는 하나 이상의 필드 이름 목록이 차트 차원 변수의 하위 집합으로 옴)을 사용하여 가능한 전체 값의 하위 집합을 만듭니다.

예 및 결과:

데이터

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

표현식이 있을 경우 Min() 함수가 표현식에서 제공한 값의 배열에서 NULL이 아닌 값을 반환해야 합니다. 따라서 이 예에서는 데이터에 NULL 값이 있으므로 함수가 표현식에서 평가된 첫 번째의 NULL이 아닌 값을 반환합니다.

예 및 결과

예	결과
`Min(UnitSales)`	2이며 `UnitSales`에서 가장 낮은 NULL이 아닌 값이기 때문입니다.

예	결과
순서의 값은 (UnitSales) 에서 판매된 단품 수에 단품 가격을 곱한 수로 계산됩니다. `Min (UnitSales*UnitPrice)`	40이며 (UnitSales)*(UnitPrice)의 가능한 모든 값을 계산한 결과 중에서 가장 낮은 NULL이 아닌 값이기 때문입니다.
`Min(UnitSales, 2)`	4이며 NULL 값 다음으로 두 번째로 낮은 값입니다.
`Min(TOTAL UnitSales)`	2이며, TOTAL 한정자는 차트 차원을 무시하고 가능한 가장 낮은 값이 발견되었음을 의미하기 때문입니다. Customer를 차원으로 사용하는 차트의 경우, TOTAL 한정자를 통해 각 고객의 최소 UnitSales가 아니라 전체 데이터 집합에 걸친 최솟값이 반환되도록 할 수 있습니다.
Customer B를 선택합니다. `Min({1} TOTAL UnitSales)`	2이며 Customer B 선택 내용과 관계 없습니다. Set Analysis 표현식 {1}이 평가할 레코드의 집합을 ALL로 정의하므로 선택 내용과 무관합니다.

예에서 사용된 데이터:

```
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 '|');
```

관련 항목:

- *FirstSortedValue - 차트 함수 (page 323)*
- *RangeMin (page 1299)*

Mode

Mode()는 **group by** 절로 정의된 표현식에서 집계된 데이터 중 가장 발생 빈도가 높은 값, 즉 모드 값을 반환합니다. **Mode()** 함수는 텍스트 값뿐 아니라 숫자 값도 반환할 수 있습니다.

구문:

```
Mode ( expr)
```

반환 데이터 유형: dual

인수

인수	설명
expr Expression	측정할 데이터가 포함된 표현식 또는 필드입니다.

제한 사항:

2개 이상의 값이 똑같은 빈도로 발생된 경우 NULL이 반환됩니다.

예 및 결과:

예제 스크립트를 앱에 추가하고 실행합니다. 그런 다음, 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.

아래의 결과 열과 동일한 결과를 얻으려면 속성 패널의 정렬 아래에서 자동을 사용자 지정으로 전환한 후 숫자순 및 사전순 정렬을 선택 취소합니다.

스크립팅 예

예	결과
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 AA가 두 번 이상 팔린 유일한 제품이기 때문입니다.

Mode - 차트 함수

Mode()는 집계된 데이터에서 가장 발생 빈도가 높은 값 즉, 모드 값을 찾습니다. **Mode()** 함수는 숫자 값뿐 아니라 텍스트 값도 처리할 수 있습니다.

구문:

```
Mode({[SetExpression] [TOTAL [<fld {,fld}>]]} expr)
```

반환 데이터 유형: dual

인수:

인수

인수	설명
expr	측정할 데이터가 포함된 표현식 또는 필드입니다.
SetExpression	기본적으로 집계 함수는 선택에 의해 정의된 사용 가능한 레코드의 집합을 집계합니다. 집합 분석 표현식으로 대체 레코드 집합을 정의할 수 있습니다.
TOTAL	**TOTAL**이 함수 인수 앞에 오는 경우, 현재 선택을 고려하되 현재 차원 값에 관련되지 않은 가능한 모든 값에 대한 계산이 실행됩니다. 즉, 차트 차원은 무시됩니다. **TOTAL [<fld {.fld}>]**(여기서 **TOTAL** 한정자 뒤에는 하나 이상의 필드 이름 목록이 차트 차원 변수의 하위 집합으로 옴)을 사용하여 가능한 전체 값의 하위 집합을 만듭니다.

예 및 결과:

데이터

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

예 및 결과

예	결과
`Mode(UnitPrice) Customer A`를 선택합니다.	15이며, `UnitSales`에서 가장 발생 빈도가 높은 값이기 때문입니다. NULL(-)을 반환합니다. 다른 값보다 발생 빈도가 높은 단일 값이 없습니다.
`Mode(Product) Customer A`를 선택합니다.	AA이며, `Product`에서 가장 발생 빈도가 높은 값이기 때문입니다. NULL(-)을 반환합니다. 다른 값보다 발생 빈도가 높은 단일 값이 없습니다.

예	결과
Mode (TOTAL UnitPrice)	15이며, TOTAL 한정자는 차트 차원을 무시하더라도 가능한 가장 발생 빈도가 높은 값이 아직 15임을 의미하기 때문입니다.
Customer B를 선택합니다. Mode({1} TOTAL UnitPrice)	15이며 Set Analysis 표현식 {1}이 선택 내용과 관계없이 평가할 레코드의 집합을 ALL로 정의하므로 선택 내용과 무관합니다.

예에서 사용된 데이터:

```
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 '|');
```

관련 항목:

- *Avg - 차트 함수 (page 389)*
- *Median - 차트 함수 (page 424)*

Only

Only()는 집계된 데이터에서 사용 가능한 결과가 하나만 있는 경우 값을 반환합니다. 레코드에 하나의 값만 포함된 경우 해당 값이 반환되고, 그렇지 않으면 NULL이 반환됩니다. 여러 레코드에 대해 평가하려면 **group by** 절을 사용하십시오. **Only()** 함수는 숫자 및 텍스트 값을 반환할 수 있습니다.

구문:

```
Only ( expr )
```

반환 데이터 유형: dual

인수

인수	설명
expr Expression	측정할 데이터가 포함된 표현식 또는 필드입니다.

예 및 결과:

예제 스크립트를 앱에 추가하고 실행합니다. 그런 다음, 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.

아래의 결과 열과 동일한 결과를 얻으려면 속성 패널의 정렬 아래에서 자동을 사용자 지정으로 전환한 후 숫자순 및 사전순 정렬을 선택 취소합니다.

```
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;
```

결과 테이블

Customer	**MyUniqIDCheck**
Astrida	1 고객 Astrida만 CustomerID가 포함된 완전한 레코드를 가지고 있기 때문입니다.

Only - 차트 함수

Only()는 집계된 데이터에서 사용 가능한 결과가 하나만 있는 경우 값을 반환합니다. 예를 들어, 단위 가격=9인 제품만 검색하는 경우 단위 가격이 9인 제품이 둘 이상 있으면 NULL을 반환합니다.

구문:

```
Only([{SetExpression}] [TOTAL [<fld {,fld}>]] expr)
```

반환 데이터 유형: dual

인수:

인수

인수	**설명**
expr	측정할 데이터가 포함된 표현식 또는 필드입니다.
SetExpression	기본적으로 집계 함수는 선택에 의해 정의된 사용 가능한 레코드의 집합을 집계합니다. 집합 분석 표현식으로 대체 레코드 집합을 정의할 수 있습니다.

인수	설명
TOTAL	**TOTAL**이 함수 인수 앞에 오는 경우, 현재 선택을 고려하되 현재 차원 값에 관련되지 않은 가능한 모든 값에 대한 계산이 실행됩니다. 즉, 차트 차원은 무시됩니다. **TOTAL [<fld {.fld}>]**(여기서 **TOTAL** 한정자 뒤에는 하나 이상의 필드 이름 목록이 차트 차원 변수의 하위 집합으로 옴)을 사용하여 가능한 전체 값의 하위 집합을 만듭니다.

샘플 데이터에 복수의 값이 존재할 수 있을 때 NULL 결과를 원할 경우 Only()를 사용합니다.

예 및 결과:

데이터

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

예 및 결과

예	결과
`Only({<UnitPrice={9}>} Product)`	BB이며, `UnitPrice`가 '9'인 유일한 `Product`이기 때문입니다.
`Only({<Product={DD}>} Customer)`	Betacab이며, 이름이 'DD'인 `Product`를 판매하는 유일한 `Customer`이기 때문입니다.
`Only({<UnitPrice={20}>} UnitSales)`	`UnitPrice`가 20일 때 `UnitSales`의 수는 2이며, `UnitPrice` =20일 때 `UnitSales`의 값이 하나뿐이기 때문입니다.
`Only({<UnitPrice={15}>} UnitSales)`	NULL이며, `UnitPrice` =15일 때 `UnitSales`의 값이 두 개이기 때문입니다.

예에서 사용된 데이터:

```
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()은 **group by** 절로 정의된 표현식에서 집계된 값의 총 합계를 계산합니다.

구문:

```
sum ( [ distinct] expr)
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
distinct	표현식 앞에 **distinct**라는 단어가 있을 경우 모든 중복 항목이 무시됩니다.
expr Expression	측정할 데이터가 포함된 표현식 또는 필드입니다.

예 및 결과:

예제 스크립트를 앱에 추가하고 실행합니다. 그런 다음, 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.

아래의 결과 열과 동일한 결과를 얻으려면 속성 패널의 정렬 아래에서 자동을 사용자 지정으로 전환한 후 숫자순 및 사전순 정렬을 선택 취소합니다.

```
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;
```

결과 테이블

Customer	MySum
Astrida	39
Betacab	9
Canutility	8

Sum - 차트 함수

Sum()은 집계 데이터 전체에서 필드 또는 표현식을 통해 지정된 값의 합계를 계산합니다.

구문:

```
Sum([{SetExpression}] [DISTINCT] [TOTAL [<fld {,fld}>]] expr)
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
expr	측정할 데이터가 포함된 표현식 또는 필드입니다.
SetExpression	기본적으로 집계 함수는 선택에 의해 정의된 사용 가능한 레코드의 집합을 집계합니다. 집합 분석 표현식으로 대체 레코드 집합을 정의할 수 있습니다.
DISTINCT	함수 인수 앞에 **DISTINCT**라는 단어가 있을 경우 해당 함수 인수의 평가 결과로 생성된 중복이 무시됩니다. *DISTINCT 한정자가 지원되기는 하지만, 일부 데이터가 누락되었음에도 전체 값이 표시되는 것으로 오해를 일으킬 수 있으므로 각별한 주의 하에 사용해야 합니다.*
TOTAL	**TOTAL**이 함수 인수 앞에 오는 경우, 현재 선택을 고려하되 현재 차원 값에 관련되지 않은 가능한 모든 값에 대한 계산이 실행됩니다. 즉, 차트 차원은 무시됩니다. **TOTAL [<fld {.fld}>]**(여기서 **TOTAL** 한정자 뒤에는 하나 이상의 필드 이름 목록이 차트 차원 변수의 하위 집합으로 옴)을 사용하여 가능한 전체 값의 하위 집합을 만듭니다.

예 및 결과:

데이터

Customer	Product	UnitSales	UnitPrice
Astrida	AA	4	16
Astrida	AA	10	15

Customer	Product	UnitSales	UnitPrice
Astrida	BB	9	9
Betacab	BB	5	10
Betacab	CC	2	20
Betacab	DD	-	25
Canutility	AA	8	15
Canutility	CC	-	19

예 및 결과

예	결과
`Sum(UnitSales)`	38. `UnitSales`에 포함된 값의 합계.
`Sum(UnitSales*UnitPrice)`	505. `UnitPrice`에 집계된 `UnitSales`를 곱한 합계.
`Sum (TOTAL UnitSales*UnitPrice)`	TOTAL 한정자는 차트 차원을 무시하고 합계가 아직 505임을 의미하므로 테이블의 모든 행과 더불어 합계에 대해 505입니다.
`Customer B`를 선택합니다. `Sum({1} TOTAL UnitSales*UnitPrice)`	505이며 Set Analysis 표현식 {1}이 선택 내용과 관계없이 평가할 레코드의 집합을 ALL로 정의하므로 선택 내용과 무관합니다.

예에서 사용된 데이터:

```
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 '|');
```

카운터 집계 함수

카운터 집계 함수는 데이터 로드 스크립트의 여러 레코드 또는 차트 차원의 여러 값에 대한 표현식의 다양한 카운트 유형을 반환합니다.

각 함수는 개요가 끝난 후에 더 자세히 설명합니다. 구문에서 함수 이름을 클릭하여 해당 함수에 대한 상세 설명에 즉시 액세스할 수도 있습니다.

데이터 로드 스크립트의 카운터 집계 함수

Count
Count()는 **group by** 절로 정의된 표현식에서 집계된 값의 수를 반환합니다.

```
Count ([distinct ] expression | * )
```

MissingCount
MissingCount()는 **group by** 절로 정의된 표현식에서 집계된 누락된 값의 수를 반환합니다.

```
MissingCount ([ distinct ] expression)
```

NullCount
NullCount()는 **group by** 절로 정의된 표현식에서 집계된 NULL 값의 수를 반환합니다.

```
NullCount ([ distinct ] expression)
```

NumericCount
NumericCount()는 **group by** 절로 정의된 표현식에서 찾은 숫자 값의 수를 반환합니다.

```
NumericCount ([ distinct ] expression)
```

TextCount
TextCount()는 **group by** 절로 정의된 표현식에서 집계된 숫자 이외 필드 값의 수를 반환합니다.

```
TextCount ([ distinct ] expression)
```

차트 표현식의 카운터 집계 함수

다음 카운터 집계 함수를 차트에서 사용할 수 있습니다.

Count
Count()는 각 차트 차원의 값, 텍스트 및 숫자의 수를 집계하는 데 사용됩니다.

```
Count - 차트 함수({[SetExpression] [DISTINCT] [TOTAL [<fld {,fld}>]]} expr)
```

MissingCount
MissingCount()는 각 차트 차원에서 누락된 값의 수를 집계하는 데 사용됩니다. 누락 값은 모두 숫자가 아닌 값입니다.

```
MissingCount - 차트 함수({[SetExpression] [DISTINCT] [TOTAL [<fld {,fld}>]]
expr)
```

NullCount
NullCount()는 각 차트 차원에서 NULL 값의 수를 집계하는 데 사용됩니다.

```
NullCount - 차트 함수({[SetExpression][DISTINCT] [TOTAL [<fld {,fld}>]]} expr)
```

NumericCount
NumericCount()는 각 차트 차원의 숫자 값 수를 집계합니다.

```
NumericCount - 차트 함수({[SetExpression] [DISTINCT] [TOTAL [<fld {,fld}>]]}
expr)
```

TextCount

TextCount()는 각 차트 차원에서 숫자가 아닌 필드 값의 수를 집계하는 데 사용됩니다.

```
TextCount - 차트 함수({[SetExpression] [DISTINCT] [TOTAL [<fld {,fld}>]]} expr)
```

Count

Count()는 **group by** 절로 정의된 표현식에서 집계된 값의 수를 반환합니다.

구문:

```
Count( [distinct ] expr)
```

반환 데이터 유형: 정수

인수:

인수

인수	설명
expr	측정할 데이터가 포함된 표현식 또는 필드입니다.
distinct	표현식 앞에 **distinct**라는 단어가 있을 경우 모든 중복 항목이 무시됩니다.

예 및 결과:

예제 스크립트를 앱에 추가하고 실행합니다. 그런 다음, 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.

아래의 결과 열과 동일한 결과를 얻으려면 속성 패널의 정렬 아래에서 자동을 사용자 지정으로 전환한 후 숫자순 및 사전순 정렬을 선택 취소합니다.

스크립팅 예

예	결과
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 차원 Customer가 시트의 테이블에 포함되어 있는 경우의 결과입니다. 아니면 OrdersByCustomer의 결과는 3, 2입니다.
이전 예에서처럼 **Temp** 테이블이 로드된 것으로 가정합니다. LOAD Count(OrderNumber) as TotalOrderNumber Resident Temp;	TotalOrderNumber 10
첫 번째 예에서처럼 **Temp** 테이블이 로드된 것으로 가정합니다. LOAD Count(distinct OrderNumber) as TotalOrderNumber Resident Temp;	TotalOrderNumber 8 OrderNumber의 두 값이 동일한 값(1과 하나의 null 값)이기 때문입니다.

Count - 차트 함수

Count()는 각 차트 차원의 값, 텍스트 및 숫자의 수를 집계하는 데 사용됩니다.

구문:

```
Count({[SetExpression] [DISTINCT] [TOTAL [<fld {,fld}>]]} expr)
```

반환 데이터 유형: 정수

인수:

인수

인수	설명
expr	측정할 데이터가 포함된 표현식 또는 필드입니다.

인수	설명
SetExpression	기본적으로 집계 함수는 선택에 의해 정의된 사용 가능한 레코드의 집합을 집계합니다. 집합 분석 표현식으로 대체 레코드 집합을 정의할 수 있습니다.
DISTINCT	함수 인수 앞에 **DISTINCT**라는 단어가 있을 경우 해당 함수 인수의 평가 결과로 생성된 중복이 무시됩니다.
TOTAL	**TOTAL**이 함수 인수 앞에 오는 경우, 현재 선택을 고려하되 현재 차원 값에 관련되지 않은 가능한 모든 값에 대한 계산이 실행됩니다. 즉, 차트 차원은 무시됩니다. **TOTAL [<fld {.fld}>]**(여기서 **TOTAL** 한정자 뒤에는 하나 이상의 필드 이름 목록이 차트 차원 변수의 하위 집합으로 옴)을 사용하여 가능한 전체 값의 하위 집합을 만듭니다.

예 및 결과:

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

다음 예에서는 달리 명시된 부분을 제외하고 모든 고객이 선택된 것으로 가정합니다.

예 및 결과

예	결과
`Count(OrderNumber)`	10이며, 빈 레코드를 포함한 모든 레코드를 계수했을 때 OrderNumber에 해당하는 값이 있을 수 있는 필드가 10개이기 때문입니다. *"0"은 빈 셀이 아니라 하나의 값으로 계산됩니다. 그러나 측정 값이 차원에 대해 0으로 집계되는 경우 해당 차원은 차트에 포함되지 않습니다.*

예	결과
`Count(Customer)`	10이며 Count는 모든 필드에서 발견된 횟수를 평가하기 때문입니다.
`Count(DISTINCT [Customer])`	4이며 Distinct 한정자를 사용하면 Count에서 고유한 발견만을 평가하기 때문입니다.
고객 Canutility가 선택된 것으로 가정할 때 `Count(OrderNumber)/Count({1} TOTAL OrderNumber)`	0.2이며, 표현식이 선택한 고객의 주문 수를 모든 고객의 주문에 대한 백분율로 반환하기 때문입니다. 이 경우 2/10입니다.
고객 Astrida 및 Canutility가 선택된 것으로 가정할 때 `Count(TOTAL <Product> OrderNumber)`	5이며, 선택한 고객에 한한 제품 주문 횟수이기 때문이고 빈 셀도 계수됩니다.

예에서 사용된 데이터:

```
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()는 **group by** 절로 정의된 표현식에서 집계된 누락된 값의 수를 반환합니다.

구문:

```
MissingCount ( [ distinct ] expr)
```

반환 데이터 유형: 정수

인수:

인수

인수	설명
expr Expression	측정할 데이터가 포함된 표현식 또는 필드입니다.
distinct	표현식 앞에 **distinct**라는 단어가 있을 경우 모든 중복 항목이 무시됩니다.

예 및 결과:

예제 스크립트를 앱에 추가하고 실행합니다. 그런 다음, 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.

아래의 결과 열과 동일한 결과를 얻으려면 속성 패널의 정렬 아래에서 자동을 사용자 지정으로 전환한 후 숫자순 및 사전순 정렬을 선택 취소합니다.

스크립팅 예

예	결과
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 두 번째 문의 결과: TotalMissingCount 3 해당 차원의 테이블에서.
이전 예에서처럼 **Temp** 테이블이 로드된 것으로 가정합니다. LOAD MissingCount(distinct OrderNumber) as TotalMissingCountDistinct Resident Temp;	TotalMissingCountDistinct 1 누락된 OrderNumber 값이 하나만 있기 때문입니다.

MissingCount - 차트 함수

MissingCount()는 각 차트 차원에서 누락된 값의 수를 집계하는 데 사용됩니다. 누락 값은 모두 숫자가 아닌 값입니다.

구문:

```
MissingCount({[SetExpression] [DISTINCT] [TOTAL [<fld {,fld}>]]} expr)
```

반환 데이터 유형: 정수

인수:

인수

인수	설명
expr	측정할 데이터가 포함된 표현식 또는 필드입니다.
SetExpression	기본적으로 집계 함수는 선택에 의해 정의된 사용 가능한 레코드의 집합을 집계합니다. 집합 분석 표현식으로 대체 레코드 집합을 정의할 수 있습니다.
DISTINCT	함수 인수 앞에 **DISTINCT**라는 단어가 있을 경우 해당 함수 인수의 평가 결과로 생성된 중복이 무시됩니다.
TOTAL	**TOTAL**이 함수 인수 앞에 오는 경우, 현재 선택을 고려하되 현재 차원 값에 관련되지 않은 가능한 모든 값에 대한 계산이 실행됩니다. 즉, 차트 차원은 무시됩니다. **TOTAL [<fld {.fld}>]**(여기서 **TOTAL** 한정자 뒤에는 하나 이상의 필드 이름 목록이 차트 차원 변수의 하위 집합으로 옴)을 사용하여 가능한 전체 값의 하위 집합을 만듭니다.

예 및 결과:

Data

Customer	Product	OrderNumber	UnitSales	Unit Price
Astrida	AA	1	4	16
Astrida	AA	7	10	15
Astrida	BB	4	9	9
Betacab	BB	6	5	10
Betacab	CC	5	2	20
Betacab	DD			25
Canutility	AA			15
Canutility	CC			19
Divadip	AA	2	4	16
Divadip	DD	3		25

예 및 결과

예	결과
`MissingCount([OrderNumber])`	3이며, OrderNumber 필드 10개 중 3개가 비었기 때문입니다.  *"0"은 빈 셀이 아니라 하나의 값으로 계산됩니다. 그러나 측정값이 차원에 대해 0으로 집계되는 경우 해당 차원은 차트에 포함되지 않습니다.*
`MissingCount ([OrderNumber])/MissingCount ({1} Total [OrderNumber])`	표현식은 선택한 고객의 미결 주문 수를 모든 고객의 미결 주문에 대한 비율로 반환하기 때문입니다. 모든 고객의 OrderNumber에서 총 3개의 누락 값이 있습니다. 따라서 Product에 대한 값이 누락된 각 Customer의 결과는 1/3입니다.

데이터 사용 예:

```
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()는 **group by** 절로 정의된 표현식에서 집계된 NULL 값의 수를 반환합니다.

구문:

```
NullCount ( [ distinct ] expr)
```

반환 데이터 유형: 정수

인수:

인수

인수	설명
expr Expression	측정할 데이터가 포함된 표현식 또는 필드입니다.
distinct	표현식 앞에 **distinct**라는 단어가 있을 경우 모든 중복 항목이 무시됩니다.

예 및 결과:

예제 스크립트를 앱에 추가하고 실행합니다. 그런 다음, 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.

아래의 결과 열과 동일한 결과를 얻으려면 속성 패널의 정렬 아래에서 자동을 사용자 지정으로 전환한 후 숫자순 및 사전순 정렬을 선택 취소합니다.

스크립팅 예

예	결과
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 두 번째 문의 결과: TotalNullCount 1 해당 차원의 테이블에서, 한 레코드에만 null 값이 포함되어 있기 때문입니다.

NullCount - 차트 함수

NullCount()는 각 차트 차원에서 NULL 값의 수를 집계하는 데 사용됩니다.

구문:

NullCount({[SetExpression][**DISTINCT**] [**TOTAL** [<fld {,fld}>]]} expr**)**

반환 데이터 유형: 정수

인수:

인수

인수	설명
expr	측정할 데이터가 포함된 표현식 또는 필드입니다.

인수	설명
set_ expression	기본적으로 집계 함수는 선택에 의해 정의된 사용 가능한 레코드의 집합을 집계합니다. 집합 분석 표현식으로 대체 레코드 집합을 정의할 수 있습니다.
DISTINCT	함수 인수 앞에 **DISTINCT**라는 단어가 있을 경우 해당 함수 인수의 평가 결과로 생성된 중복이 무시됩니다.
TOTAL	**TOTAL**이 함수 인수 앞에 오는 경우, 현재 선택을 고려하되 현재 차원 값에 관련되지 않은 가능한 모든 값에 대한 계산이 실행됩니다. 즉, 차트 차원은 무시됩니다. **TOTAL [<fld {.fld}>]**(여기서 **TOTAL** 한정자 뒤에는 하나 이상의 필드 이름 목록이 차트 차원 변수의 하위 집합으로 옴)을 사용하여 가능한 전체 값의 하위 집합을 만듭니다.

예 및 결과:

예 및 결과

예	결과
NullCount ([OrderNumber])	1이며, 인라인 **LOAD** 문 내의 NullInterpret을 사용하여 Null 값을 도입했기 때문입니다.

데이터 사용 예:

```
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()는 **group by** 절로 정의된 표현식에서 찾은 숫자 값의 수를 반환합니다.

구문:

```
NumericCount ( [ distinct ] expr)
```

반환 데이터 유형: 정수

인수:

인수

인수	설명
expr Expression	측정할 데이터가 포함된 표현식 또는 필드입니다.
distinct	표현식 앞에 **distinct**라는 단어가 있을 경우 모든 중복 항목이 무시됩니다.

예 및 결과:

예제 스크립트를 앱에 추가하고 실행합니다. 그런 다음, 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.

아래의 결과 열과 동일한 결과를 얻으려면 속성 패널의 정렬 아래에서 자동을 사용자 지정으로 전환한 후 숫자순 및 사전순 정렬을 선택 취소합니다.

스크립팅 예

예	결과
`LOAD NumericCount(OrderNumber) as TotalNumericCount Resident Temp;`	두 번째 문의 결과: TotalNumericCount 7 해당 차원의 테이블에서.
이전 예에서처럼 **Temp** 테이블이 로드된 것으로 가정합니다. `LOAD NumericCount(distinct OrderNumber) as TotalNumericCountDistinct Resident Temp;`	TotalNumericCountDistinct 6 다른 항목을 복제하는 OrderNumber가 하나 있으므로 결과는 복제되지 않은 6입니다.

```
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;
```

결과 테이블

Customer	NumericCountByCustomer
Astrida	3
Betacab	2
Canutility	0
Divadip	2

NumericCount - 차트 함수

NumericCount()는 각 차트 차원의 숫자 값 수를 집계합니다.

구문:

```
NumericCount({[SetExpression] [DISTINCT] [TOTAL [<fld {,fld}>]]} expr)
```

반환 데이터 유형: 정수

인수:

인수

인수	설명
expr	측정할 데이터가 포함된 표현식 또는 필드입니다.
set_ expression	기본적으로 집계 함수는 선택에 의해 정의된 사용 가능한 레코드의 집합을 집계합니다. 집합 분석 표현식으로 대체 레코드 집합을 정의할 수 있습니다.
DISTINCT	함수 인수 앞에 **DISTINCT**라는 단어가 있을 경우 해당 함수 인수의 평가 결과로 생성된 중복이 무시됩니다.

인수	설명
TOTAL	**TOTAL**이 함수 인수 앞에 오는 경우, 현재 선택을 고려하되 현재 차원 값에 관련되지 않은 가능한 모든 값에 대한 계산이 실행됩니다. 즉, 차트 차원은 무시됩니다. **TOTAL [<fld {.fld}>]**(여기서 **TOTAL** 한정자 뒤에는 하나 이상의 필드 이름 목록이 차트 차원 변수의 하위 집합으로 옴)을 사용하여 가능한 전체 값의 하위 집합을 만듭니다.

예 및 결과:

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

다음 예에서는 달리 명시된 부분을 제외하고 모든 고객이 선택된 것으로 가정합니다.

예 및 결과

예	결과
`NumericCount ([OrderNumber])`	7이며, OrderNumber 필드 10개 중 3개가 비었기 때문입니다. *"0"은 빈 셀이 아니라 하나의 값으로 계산됩니다. 그러나 측정값이 차원에 대해 0으로 집계되는 경우 해당 차원은 차트에 포함되지 않습니다.*
`NumericCount ([Product])`	0이며 모든 제품 이름이 텍스트이기 때문입니다. 일반적으로 이는 텍스트 필드에 숫자 내용이 입력되지 않았는지 확인하는 데 사용할 수 있습니다.

예	결과
NumericCount (DISTINCT [OrderNumber])/Count (DISTINCT [OrderNumber)]	고유 숫자 주문 번호의 수를 계수하고 이를 숫자 및 숫자가 아닌 주문 번호의 수로 나눕니다. 모든 필드 값이 숫자라면 결과가 1이 됩니다. 일반적으로 이를 사용하여 모든 필드 값이 숫자인지 확인할 수 있습니다. 예에서는 8개의 숫자 및 숫자가 아닌 고유 OrderNumber에서 고유 숫자 값이 7개이므로 표현식은 0.875를 반환합니다.

데이터 사용 예:

```
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()는 **group by** 절로 정의된 표현식에서 집계된 숫자 이외 필드 값의 수를 반환합니다.

구문:

```
TextCount ( [ distinct ] expr)
```

반환 데이터 유형: 정수

인수:

인수

인수	설명
expr Expression	측정할 데이터가 포함된 표현식 또는 필드입니다.
distinct	표현식 앞에 **distinct**라는 단어가 있을 경우 모든 중복 항목이 무시됩니다.

예 및 결과:

예제 스크립트를 앱에 추가하고 실행합니다. 그런 다음, 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.

아래의 결과 열과 동일한 결과를 얻으려면 속성 패널의 정렬 아래에서 자동을 사용자 지정으로 전환한 후 숫자순 및 사전순 정렬을 선택 취소합니다.

```
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;
```

결과 테이블

Customer	**ProductTextCount**
Astrida	3
Betacab	3
Canutility	2
Divadip	2

```
LOAD Customer,TextCount(OrderNumber) as OrderNumberTextCount Resident Temp Group By Customer;
```

결과 테이블

Customer	**OrderNumberTextCount**
Astrida	0
Betacab	1
Canutility	2
Divadip	0

TextCount - 차트 함수

TextCount()는 각 차트 차원에서 숫자가 아닌 필드 값의 수를 집계하는 데 사용됩니다.

구문:

```
TextCount({[SetExpression] [DISTINCT] [TOTAL [<fld {,fld}>]]} expr)
```

반환 데이터 유형: 정수

인수:

인수

인수	설명
expr	측정할 데이터가 포함된 표현식 또는 필드입니다.
SetExpression	기본적으로 집계 함수는 선택에 의해 정의된 사용 가능한 레코드의 집합을 집계합니다. 집합 분석 표현식으로 대체 레코드 집합을 정의할 수 있습니다.
DISTINCT	함수 인수 앞에 **DISTINCT**라는 단어가 있을 경우 해당 함수 인수의 평가 결과로 생성된 중복이 무시됩니다.
TOTAL	**TOTAL**이 함수 인수 앞에 오는 경우, 현재 선택을 고려하되 현재 차원 값에 관련되지 않은 가능한 모든 값에 대한 계산이 실행됩니다. 즉, 차트 차원은 무시됩니다. **TOTAL [<fld {.fld}>]**(여기서 **TOTAL** 한정자 뒤에는 하나 이상의 필드 이름 목록이 차트 차원 변수의 하위 집합으로 옴)을 사용하여 가능한 전체 값의 하위 집합을 만듭니다.

예 및 결과:

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

예 및 결과

예	결과
TextCount ([Product])	10이며, Product의 필드 10개 모두가 텍스트이기 때문입니다. *"0"은 빈 셀이 아니라 하나의 값으로 계산됩니다. 그러나 측정값이 차원에 대해 0으로 집계되는 경우 해당 차원은 차트에 포함되지 않습니다. 빈 셀은 텍스트가 없는 것으로 평가되어 TextCount에서 계수되지 않습니다.*
TextCount ([OrderNumber])	3이며, 빈 셀이 계수되었기 때문입니다. 일반적으로 이는 숫자 필드에 텍스트 값이 입력되지 않았는지 또는 0이 아닌지 확인하는 데 사용됩니다.
TextCount (DISTINCT [Product])/Count ([Product)]	Product의 모든 고유 텍스트 값의 수(4)를 계수하고 이를 Product의 값의 총 수(10)로 나눕니다. 결과는 0.4입니다.

데이터 사용 예:

```
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 '|');
```

재무 집계 함수

이 섹션에서는 납입 및 현금 흐름과 관련된 재무 작업을 위한 집계 함수에 대해 설명합니다.

각 함수는 개요가 끝난 후에 더 자세히 설명합니다. 구문에서 함수 이름을 클릭하여 해당 함수에 대한 상세 설명에 즉시 액세스할 수도 있습니다.

데이터 로드 스크립트의 재무 집계 함수

IRR

IRR()은 group by 절로 정의된 여러 레코드에서 반복되는 표현식에서 숫자로 표현된 일련의 현금 흐름에 대해 집계된 내부 수익률을 반환합니다.

```
IRR (expression)
```

XIRR

XIRR()은 group by 절로 정의된 여러 레코드에서 반복되는 **pmt** 및 **date**에서 숫자 쌍으로 표현된 현금 흐름 일정(주기적일 필요는 없음)에 대해 집계된 내부 수익률(연간)을 반환합니다. 모든 납입금은 1년 365일을 기준으로 할인됩니다.

```
XIRR (valueexpression, dateexpression )
```

NPV

NPV() 스크립트 함수는 기간별로 정렬된 여러 값과 할인율을 사용합니다. 유입(수입)은 양수이고 유출(미래 지불)은 이러한 계산에서 음수 값으로 가정됩니다. 이는 각 기간이 끝날 때 발생합니다.

```
NPV (rate, expression)
```

XNPV

XNPV()은 **pmt** 및 **date**에서 숫자 쌍으로 표현된 현금 흐름 일정(반드시 주기적일 필요는 없음)에 대해 집계된 순 현재 가치를 반환합니다. 모든 납입금은 1년 365일을 기준으로 할인됩니다.

```
XNPV (rate, valueexpression, dateexpression)
```

차트 표현식의 재무 집계 함수

이러한 재무 집계 함수는 차트에서 사용할 수 있습니다.

IRR

IRR()은 차트 차원에서 반복되는 **value**를 통해 지정된 표현식의 숫자로 표현된 일련의 현금 흐름에 대해 집계된 내부 수익률을 반환합니다.

```
IRR - 차트 함수([TOTAL] [<fld {,fld}>]] value)
```

NPV

NPV()는 기간별 **discount_rate** 및 차트 차원에서 반복되는 **value**의 숫자로 표현된 일련의 미래 납입(음수)과 수입(양수)을 기준으로 집계된 투자의 순 현재 가치를 반환합니다. 납입 및 수입은 각 기간 말에 발생한다고 가정합니다.

```
NPV - 차트 함수([TOTAL] [<fld {,fld}>]] discount_rate, value)
```

XIRR

XIRR()은 차트 차원에서 반복되는 **pmt** 및 **date**를 통해 지정된 표현식의 숫자 쌍으로 표현된 현금 흐름 일정(주기적일 필요는 없음)에 대해 집계된 내부 수익률(연간)을 반환합니다. 모든 납입금은 1년 365일을 기준으로 할인됩니다.

```
XIRR - 차트 함수([TOTAL] [<fld {,fld}>]] pmt, date)
```

XNPV

XNPV()는 차트 차원에서 반복되는 **pmt** 및 **date**를 통해 지정된 표현식의 숫자 쌍으로 표현된 현금 흐름 일정(주기적일 필요는 없음)에 대해 집계된 순 현재 가치를 반환합니다. 모든 납입금은 1년 365일을 기준으로 할인됩니다.

```
XNPV - 차트 함수([TOTAL] [<fld{,fld}>]] discount_rate, pmt, date)
```

IRR

IRR()은 group by 절로 정의된 여러 레코드에서 반복되는 표현식에서 숫자로 표현된 일련의 현금 흐름에 대해 집계된 내부 수익률을 반환합니다.

이 현금 흐름은 연금에 대한 것이므로 균일할 필요가 없습니다. 하지만 매월 또는 매년과 같이 일정한 간격으로 현금 흐름이 일어나야 합니다. 내부 수익률은 정기적인 기간에 일어나는 납입(음수) 및 수입(양수)으로 구성된 투자에 대해 수급되는 이자율입니다. 이 함수를 계산하려면 최소 하나 이상의 양수와 하나 이상의 음수 값이 필요합니다.

이 함수는 내부 반환율(IRR)을 계산하기 위해 간소화된 버전의 Newton 방법을 사용합니다.

구문:

```
IRR(value)
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
value	측정할 데이터가 포함된 표현식 또는 필드입니다.

제한 사항:

텍스트 값, NULL 값, 누락된 값은 무시됩니다.

예 및 결과:

예제 스크립트를 앱에 추가하고 실행합니다. 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.

예 및 결과:

예 및 결과

예	연도	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 - 차트 함수

IRR()은 차트 차원에서 반복되는 **value**를 통해 지정된 표현식의 숫자로 표현된 일련의 현금 흐름에 대해 집계된 내부 수익률을 반환합니다.

이 현금 흐름은 연금에 대한 것이므로 균일할 필요가 없습니다. 하지만 매월 또는 매년과 같이 일정한 간격으로 현금 흐름이 일어나야 합니다. 내부 수익률은 정기적인 기간에 일어나는 납입(음수) 및 수입(양수)으로 구성된 투자에 대해 수급되는 이자율입니다. 이 함수를 계산하려면 최소 하나 이상의 양수와 하나 이상의 음수 값이 필요합니다.

이 함수는 내부 반환율(IRR)을 계산하기 위해 간소화된 버전의 Newton 방법을 사용합니다.

구문:

```
IRR([TOTAL [<fld {,fld}>]] value)
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
value	측정할 데이터가 포함된 표현식 또는 필드입니다.
TOTAL	**TOTAL**이 함수 인수 앞에 오는 경우, 현재 선택을 고려하되 현재 차원 값에 관련되지 않은 가능한 모든 값에 대한 계산이 실행됩니다. 즉, 차트 차원은 무시됩니다. **TOTAL [<fld {.fld}>]**(여기서 **TOTAL** 한정자 뒤에는 하나 이상의 필드 이름 목록이 차트 차원 변수의 하위 집합으로 옴)을 사용하여 가능한 전체 값의 하위 집합을 만듭니다.

제한 사항:

집계 함수의 매개 변수는 이러한 내부 집계에 **TOTAL** 한정자가 포함되어 있지 않는 한 다른 집계 함수를 포함하지 않아야 합니다. 고급 중첩 집계가 필요한 경우는 고급 함수 **Aggr**을 지정된 차원과 함께 사용하십시오.

텍스트 값, NULL 값, 누락된 값은 무시됩니다.

예 및 결과:

예 및 결과

예	결과
`IRR (Payments)`	0.1634 납입은 본질적으로 예를 들어 매월과 같이 주기적이 될 것으로 가정됩니다. *날짜 필드는 납입이 이루어진 날짜를 제공하는 한, 납입이 비주기적으로 이루어질 수 있는 XIRR 예에서 사용됩니다.*

예에서 사용된 데이터:

```
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 '|');
```

관련 항목:

- *XIRR - 차트 함수 (page 371)*
- *Aggr - 차트 함수 (page 524)*

NPV

NPV() 스크립트 함수는 기간별로 정렬된 여러 값과 할인율을 사용합니다. 유입(수입)은 양수이고 유출(미래 지불)은 이러한 계산에서 음수 값으로 가정됩니다. 이는 각 기간이 끝날 때 발생합니다.

순 현재 가치(NPV)는 미래 현금 흐름 스트림의 현재 총 가치를 계산하는 데 사용됩니다. NPV를 계산하려면 각 기간의 미래 현금 흐름을 추정하고 정확한 할인율을 확인해야 합니다. **NPV()** 스크립트 함수는 할인율과 기간별로 정렬된 여러 값을 사용합니다. 유입(수입)은 양수이고 유출(미래 지불)은 이러한 계산에서 음수 값으로 가정됩니다. 이는 각 기간이 끝날 때 발생합니다.

구문:

```
NPV(discount_rate, value)
```

반환 데이터 유형: 숫자 기본적으로 결과는 통화로 서식이 지정됩니다.

순 현재 가치를 계산하는 공식은 다음과 같습니다.

$$NPV = \sum_{t=1}^{n} \frac{R_t}{(1+i)^t}$$

설명:

- R_t = 단일 기간 t 동안의 순 현금 유입-유출
- i = 대체 투자로 얻을 수 있는 할인율 또는 수익
- t = 타이머 기간 수

인수

인수	설명
discount_rate	**discount_rate**는 적용된 할인율입니다. 값이 0.1이면 10% 할인율을 나타냅니다.
value	이 필드는 기간별로 정렬된 여러 기간에 대한 값을 보유합니다. 첫 번째 값은 기간 1이 끝날 때의 현금 흐름으로 가정하는 식입니다.

제한 사항:

NPV() 함수에는 다음과 같은 제한 사항이 있습니다.

- 텍스트 값, NULL 값, 누락된 값은 무시됩니다.
- 현금 흐름 값은 기간 오름차순이어야 합니다.

사용 시기

NPV()는 프로젝트 수익성을 확인하고 기타 측정값을 도출하는 데 사용되는 재무 함수입니다. 이 함수는 현금 흐름을 원시 데이터로 사용할 수 있는 경우에 유용합니다.

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

예 1 – 단일 지불(스크립트)

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- CashFlow라는 테이블에 로드되는 한 프로젝트의 데이터 집합 및 한 기간 동안의 해당 현금 흐름.
- NPV라는 테이블에서 프로젝트의 NPV 필드를 계산하는 데 사용되는 CashFlow 테이블의 Resident LOAD.
- NPV 계산에 사용되는 하드 코딩된 할인율 10%.
- 프로젝트에 대한 모든 지불을 그룹화하는 데 사용되는 Group By 문.

로드 스크립트

```
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;
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- PrjId
- NPV

결과 테이블

PrjId	NPV
1	$909.09

한 기간이 끝날 때 기간당 10%의 할인율로 $1000의 단일 지불을 받을 경우 NPV는 $1000를 (1 + 할인율)로 나눈 값과 같습니다. 유효 NPV는 $909.09입니다.

예 2 – 여러 지불(스크립트)

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- CashFlow라는 테이블에 로드되는 한 프로젝트의 데이터 집합 및 여러 기간 동안의 해당 현금 흐름.
- NPV라는 테이블에서 프로젝트의 NPV 필드를 계산하는 데 사용되는 CashFlow 테이블의 Resident LOAD.
- NPV 계산에 사용되는 하드 코딩된 할인율 10%(0.1).
- 프로젝트에 대한 모든 지불을 그룹화하는 데 사용되는 Group By 문.

로드 스크립트

```
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;
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- PrjId
- NPV

결과 테이블

PrjId	NPV
1	$1735.54

기간당 10%의 할인율로 두 기간의 끝에 받을 $1000 지불의 경우 유효 NPV는 $1735.54입니다.

예 3 – 여러 지불(스크립트)

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Project라는 테이블에 로드되는 두 프로젝트의 할인율.
- 프로젝트 ID 및 기간 ID별로 각 프로젝트에 대한 여러 기간에 대한 현금 흐름. 이 기간 ID는 데이터가 정렬되지 않은 경우 레코드를 정렬하는 데 사용할 수 있습니다.
- 임시 테이블 tmpNPV를 만드는 NoConcatenate, Resident LOAD 및 Left Join 함수 조합. 이 테이블은 Project 및 CashFlow 테이블의 레코드를 하나의 플랫 테이블로 결합합니다. 이 테이블에는 각 기간에 대해 반복되는 할인율이 있습니다.

- NPV라는 테이블의 각 프로젝트에 대한 NPV 필드를 계산하는 데 사용되는 tmpNPV 테이블의 Resident LOAD.
- 각 프로젝트와 연결된 단일 값 할인율입니다. 이는 only() 함수를 사용하여 검색되며 각 프로젝트의 NPV 계산에 사용됩니다.
- 프로젝트 ID별로 각 프로젝트에 대한 모든 지불을 그룹화하는 데 사용되는 Group By 문.

합성 데이터나 중복 데이터가 데이터 모델에 로드되는 것을 방지하기 위해 tmpNPV 테이블은 스크립트 끝에서 삭제됩니다.

로드 스크립트

```
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;
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- PrjId
- NPV

결과 테이블

PrjId	NPV
1	$2486.85
2	$2042.12

프로젝트 ID 1은 기간당 10%의 할인율로 세 기간이 끝날 때 $1000를 지불할 것으로 예상합니다. 따라서 유효 NPV는 $2486.85입니다.

프로젝트 ID 2는 15%의 할인율로 네 기간에 걸쳐 $500의 두 번 지불과 $1000의 추가 두 번 지불을 예상합니다. 따라서 유효 NPV는 $2042.12입니다.

예 4 – 프로젝트 수익성 예(스크립트)

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Project라는 테이블에 로드된 두 프로젝트에 대한 할인율 및 초기 투자(기간 0).
- 프로젝트 ID 및 기간 ID별로 각 프로젝트에 대한 여러 기간에 대한 현금 흐름. 이 기간 ID는 데이터가 정렬되지 않은 경우 레코드를 정렬하는 데 사용할 수 있습니다.
- 임시 테이블 tmpNPV를 만드는 NoConcatenate, Resident LOAD 및 Left Join 함수 조합. 이 테이블은 Project 및 CashFlow 테이블의 레코드를 하나의 플랫 테이블로 결합합니다. 이 테이블에는 각 기간에 대해 반복되는 할인율이 있습니다.
- only() 함수를 사용하여 검색하고 각 프로젝트에 대한 NPV 계산에 사용되는 각 프로젝트와 연결된 단일 값 할인율.
- NPV라는 테이블의 각 프로젝트에 대한 NPV 필드를 계산하는 데 사용되는 tmpNPV 테이블의 Resident LOAD.
- 프로젝트 수익성 지수를 계산하기 위해 NPV를 각 프로젝트의 초기 투자로 나누는 추가 필드 만들기.
- 프로젝트 ID별로 그룹화하는 group by 문. 각 프로젝트에 대한 모든 지불을 그룹화하는 데 사용됩니다.

합성 데이터나 중복 데이터가 데이터 모델에 로드되는 것을 방지하기 위해 tmpNPV 테이블은 스크립트 끝에서 삭제됩니다.

로드 스크립트

```
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;
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- PrjId
- NPV

다음 측정값을 만듭니다.

```
=only(Profitability_Index)
```

결과 테이블

PrjId	NPV	=only(Profitability_Index)
1	$87039.82	0.87
2	$123513.71	1.24

프로젝트 ID 1의 유효 NPV는 $87039.82이고 초기 투자 금액은 $100000입니다. 따라서 수익성 지수는 0.87입니다. 1보다 작기 때문에 프로젝트에 수익성이 없습니다.

프로젝트 ID 2의 유효 NPV는 $123513.71이고 초기 투자 금액은 $100000입니다. 따라서 수익성 지수는 1.24입니다. 1보다 크므로 프로젝트에 수익성이 있습니다.

NPV - 차트 함수

NPV()는 기간별 **discount_rate** 및 차트 차원에서 반복되는 **value**의 숫자로 표현된 일련의 미래 납입(음수)과 수입(양수)을 기준으로 집계된 투자의 순 현재 가치를 반환합니다. 납입 및 수입은 각 기간 말에 발생한다고 가정합니다.

구문:

```
NPV([TOTAL [<fld {,fld}>]] discount_rate, value)
```

반환 데이터 유형: 숫자 기본적으로 결과는 통화로 서식이 지정됩니다.

인수:

인수

인수	설명
discount_rate	**discount_rate**는 적용된 할인율입니다.
value	측정할 데이터가 포함된 표현식 또는 필드입니다.
TOTAL	**TOTAL**이 함수 인수 앞에 오는 경우, 현재 선택을 고려하되 현재 차원 값에 관련되지 않은 가능한 모든 값에 대한 계산이 실행됩니다. 즉, 차트 차원은 무시됩니다. **TOTAL [<fld {.fld}>]**(여기서 **TOTAL** 한정자 뒤에는 하나 이상의 필드 이름 목록이 차트 차원 변수의 하위 집합으로 옴)을 사용하여 가능한 전체 값의 하위 집합을 만듭니다. **TOTAL** 한정자 뒤에는 꺾쇠 괄호로 묶인 하나 이상의 필드 이름 목록이 올 수 있습니다. 이러한 필드 이름은 차트 차원 변수의 하위 집합이어야 합니다. 이 경우 나열된 차원을 제외하고 모든 차트 차원 변수를 무시하면서 계산이 실행됩니다. 즉, 나열된 차원 필드 내 필드 값의 각 조합에 대해 하나의 값이 반환됩니다. 또한 현재 차트 내의 차원이 아닌 필드가 목록에 포함될 수 있습니다. 이는 차원 필드가 고정되지 않은 그룹 차원의 경우에 유용할 수 있습니다. 그룹 내의 모든 변수를 나열할 경우 드릴다운 수준이 변화할 때 함수가 작동하게 됩니다.

제한 사항:

내부 집계에 **TOTAL** 한정자가 포함된 경우를 제외하고, **discount_rate** 및 **value**에 집계 함수가 포함되지 않아야 합니다. 고급 중첩 집계가 필요한 경우는 고급 함수 **Aggr**을 지정된 차원과 함께 사용하십시오.

텍스트 값, NULL 값, 누락된 값은 무시됩니다.

예 및 결과:

예 및 결과

예	결과
NPV(Discount, Payments)	-$540.12

예에서 사용된 데이터:

```
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 '|');
```

관련 항목:

- *XNPV - 차트 함수 (page 380)*
- *Aggr - 차트 함수 (page 524)*

XIRR

XIRR()은 group by 절로 정의된 여러 레코드에서 반복되는 **pmt** 및 **date**에서 숫자 쌍으로 표현된 현금 흐름 일정(주기적일 필요는 없음)에 대해 집계된 내부 수익률(연간)을 반환합니다. 모든 납입금은 1년 365일을 기준으로 할인됩니다.

Qlik의 XIRR 기능(**XIRR()** 및 **RangeXIRR()** 함수)은 다음 등식을 통해 Rate 값을 구하여 올바른 XIRR 값을 확인합니다.

```
XNPV(Rate, pmt, date) = 0
```

이 등식은 Newton 방법의 간소화된 버전을 사용하여 구합니다.

구문:

```
XIRR(pmt, date )
```

반환 데이터 유형: 숫자

인수

인수	설명
pmt	지불. **date**에 제공된 지불 일정에 해당하는 현금 흐름을 포함하는 표현식 또는 필드.
date	**pmt**에 지정된 현금 흐름 지급에 해당하는 날짜의 일정을 포함하는 표현식 또는 필드입니다.

이 함수를 사용할 때 다음 제한 사항이 적용됩니다.

- 데이터 쌍의 한쪽 또는 양쪽에 텍스트 값, NULL 값, 누락된 값이 있으면 전체 데이터 쌍이 무시됩니다.
- 이 함수는 하나 이상의 유효한 음성 지불과 하나 이상의 유효한 양성 지불(해당 유효 날짜 포함)이 필요합니다. 이러한 결제가 제공되지 않으면 NULL 값이 반환됩니다.

다음 항목은 이 함수를 사용하는 데 도움이 될 수 있습니다.

- *XNPV (page 374)*: 이 함수를 사용하여 현금 흐름 일정에 대해 집계된 순 현재 가치를 계산합니다.
- *RangeXIRR (page 1317)*: **RangeXIRR()**은 **XIRR()** 함수와 동등한 범위 함수입니다.

Qlik Sense 클라이언트 관리의 다양한 버전 간에는 이 함수에서 사용하는 기본 알고리즘에 변형이 있습니다. 알고리즘의 최근 업데이트에 대한 자세한 내용은 지원 문서 XIRR 함수 수정 및 업데이트를 참조하십시오.

예

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 일련의 현금 흐름에 대한 트랜잭션 데이터입니다.
- **XIRR()** 함수를 사용하여 이러한 현금 흐름에 대한 내부 연간 수익률을 계산합니다.

로드 스크립트

```
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;
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- Year
- XIRR2013

결과 테이블

연도	XIRR2013
2013	0.5385

XIRR 반환 값 해석

XIRR 기능은 일반적으로 투자를 분석하는 데 사용되며, 처음에는 나가는(음성) 지불이 발생하고 나중에 일련의 작은 소득(양성) 지불이 발생합니다. 다음은 하나의 음성 지불과 하나의 양성 지불만 있는 간단한 예입니다.

```
Cashflow:
LOAD * inline [
Date|Payments
2023-01-01|-100
2024-01-01|110
] (delimiter is '|');
```

초기에 100을 지불하고 정확히 1년 후에 110을 가져옵니다. 이는 연 10%의 수익률을 의미합니다. `XIRR (Payments, Date)`는 0.1 값을 반환합니다.

XIRR 기능의 반환 값은 양수 또는 음수일 수 있습니다. 투자의 경우, 음수 결과는 투자가 손실임을 나타냅니다. 손익 금액은 지불 필드에서 합계를 집계하여 간단하게 계산할 수 있습니다.

위의 예에서 우리는 1년 동안 돈을 빌려주고 있습니다. 수익률은 이자라고 생각할 수 있습니다. 트랜잭션의 반대편에 있을 때 XIRR의 기능을 사용하는 것도 가능합니다(예를 들어, 대출 기관이 아닌 차용인인 경우).

다음 예를 생각해봅니다.

```
Cashflow:
LOAD * inline [
Date|Payments
2023-01-01|100
2024-01-01|-110
] (delimiter is '|');
```

이는 첫 번째 예와 동일하지만 반대의 경우입니다. 여기서 우리는 1년 동안 100을 빌리고 10%의 이자로 갚습니다. 이 예에서 XIRR 계산은 첫 번째 예와 동일한 값인 0.1(10%)을 반환합니다.

첫 번째 예에서는 10의 이익을 얻었고 두 번째 예에서는 10의 손실을 경험했지만, 두 예에서 XIRR 기능의 반환 값은 양수입니다. 이는 XIRR 기능이 트랜잭션의 어느 쪽에 있든 관계없이 트랜잭션의 숨겨진 이자를 계산하기 때문입니다.

여러 솔루션의 제한 사항

Qlik의 XIRR 기능은 `Rate` 값이 해결되는 다음 방정식으로 정의됩니다.

```
XNPV(Rate, pmt, date) = 0
```

때때로 이 방정식에 하나 이상의 해가 있을 수 있습니다. 이는 "다중 IRR 문제"로 알려져 있으며 비정규 현금 흐름 스트림(비규칙적 현금 흐름이라고도 함)으로 인해 발생합니다. 다음 로드 스크립트는 이에 대한 예를 보여 줍니다.

```
Cashflow:
LOAD * inline [
Date|Payments
2021-01-01|-200
2022-01-01|500
2023-01-01|-250
] (delimiter is '|');
```

이 예에는 하나의 음성 솔루션과 하나의 양성 솔루션이 있습니다(Rate = -0.3 및 Rate = 0.8). **XIRR()**은 0.8을 반환합니다.

Qlik의 XIRR 기능은 솔루션을 검색할 때 Rate = 0에서 시작하여 솔루션을 찾을 때까지 단계적으로 속도를 높입니다. 둘 이상의 양성 솔루션이 있는 경우 가장 먼저 만나는 솔루션을 반환합니다. 양성 솔루션을 찾을 수 없으면 Rate를 다시 0으로 다시 설정하고 음성 방향에서 솔루션 검색을 시작합니다.

"정상적인" 현금 흐름 스트림에는 단 하나의 솔루션만 있음이 보장됩니다. "정상적인" 현금 흐름 스트림은 동일한 부호(양수 또는 음수)를 가진 모든 지불이 연속적인 그룹에 있음을 의미합니다.

관련 항목:

- *XNPV (page 374)*
- *RangeXIRR (page 1317)*
- XIRR 함수 수정 및 업데이트

XIRR - 차트 함수

XIRR()은 차트 차원에서 반복되는 **pmt** 및 **date**를 통해 지정된 표현식의 숫자 쌍으로 표현된 현금 흐름 일정(주기적일 필요는 없음)에 대해 집계된 내부 수익률(연간)을 반환합니다. 모든 납입금은 1년 365일을 기준으로 할인됩니다.

Qlik의 XIRR 기능(**XIRR()** 및 **RangeXIRR()** 함수)은 다음 등식을 통해 Rate 값을 구하여 올바른 XIRR 값을 확인합니다.

```
XNPV(Rate, pmt, date) = 0
```

이 등식은 Newton 방법의 간소화된 버전을 사용하여 구합니다.

구문:

```
XIRR([TOTAL [<fld {,fld}>]] pmt, date)
```

반환 데이터 유형: 숫자

인수

인수	설명
pmt	지불. **date**에 제공된 지불 일정에 해당하는 현금 흐름을 포함하는 표현식 또는 필드.
date	**pmt**에 지정된 현금 흐름 지급에 해당하는 날짜의 일정을 포함하는 표현식 또는 필드입니다.

인수	설명
TOTAL	**TOTAL**이 함수 인수 앞에 오는 경우, 현재 선택을 고려하되 현재 차원 값에 관련되지 않은 가능한 모든 값에 대한 계산이 실행됩니다. 즉, 차트 차원은 무시됩니다. **TOTAL [<fld {.fld}>]**(여기서 **TOTAL** 한정자 뒤에는 하나 이상의 필드 이름 목록이 차트 차원 변수의 하위 집합으로 옴)을 사용하여 가능한 전체 값의 하위 집합을 만듭니다.

이 함수를 사용할 때 다음 제한 사항이 적용됩니다.

- 내부 집계에 **TOTAL** 한정자가 포함된 경우를 제외하고, **pmt** 및 **date**에 집계 함수가 포함되지 않아야 합니다. 고급 중첩 집계가 필요한 경우는 고급 함수 **Aggr**을 지정된 차원과 함께 사용하십시오.
- 데이터 쌍의 한쪽 또는 양쪽에 텍스트 값, NULL 값, 누락된 값이 있으면 전체 데이터 쌍이 무시됩니다.
- 이 함수는 하나 이상의 유효한 음성 지불과 하나 이상의 유효한 양성 지불(해당 유효 날짜 포함)이 필요합니다. 이러한 결제가 제공되지 않으면 NULL 값이 반환됩니다.

다음 항목은 이 함수를 사용하는 데 도움이 될 수 있습니다.

- *XNPV - 차트 함수 (page 380)*: 이 함수를 사용하여 현금 흐름 일정에 대해 집계된 순 현재 가치를 계산합니다.
- *RangeXIRR (page 1317)*: **RangeXIRR()**은 **XIRR()** 함수와 동등한 범위 함수입니다.

Qlik Sense 클라이언트 관리의 다양한 버전 간에는 이 함수에서 사용하는 기본 알고리즘에 변형이 있습니다. 알고리즘의 최근 업데이트에 대한 자세한 내용은 지원 문서 XIRR 함수 수정 및 업데이트를 참조하십시오.

예

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 현금 흐름 트랜잭션이 포함된 데이터 집합.
- Cashflow라는 테이블에 저장된 정보.

로드 스크립트

```
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 '|');
```

결과

다음과 같이 하십시오.

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 계산을 측정값으로 추가합니다.

```
=XIRR(Payments, Date)
```

결과 테이블

=XIRR(Payments, Date)
0.5385

XIRR 반환 값 해석

XIRR 기능은 일반적으로 투자를 분석하는 데 사용되며, 처음에는 나가는(음성) 지불이 발생하고 나중에 일련의 작은 소득(양성) 지불이 발생합니다. 다음은 하나의 음성 지불과 하나의 양성 지불만 있는 간단한 예입니다.

```
Cashflow:
LOAD * inline [
Date|Payments
2023-01-01|-100
2024-01-01|110
] (delimiter is '|');
```

초기에 100을 지불하고 정확히 1년 후에 110을 가져옵니다. 이는 연 10%의 수익률을 의미합니다. `XIRR (Payments, Date)`는 0.1 값을 반환합니다.

XIRR 기능의 반환 값은 양수 또는 음수일 수 있습니다. 투자의 경우, 음수 결과는 투자가 손실임을 나타냅니다. 손익 금액은 지불 필드에서 합계를 집계하여 간단하게 계산할 수 있습니다.

위의 예에서 우리는 1년 동안 돈을 빌려주고 있습니다. 수익률은 이자라고 생각할 수 있습니다. 트랜잭션의 반대편에 있을 때 XIRR의 기능을 사용하는 것도 가능합니다(예를 들어, 대출 기관이 아닌 차용인인 경우).

다음 예를 생각해봅니다.

```
Cashflow:
LOAD * inline [
Date|Payments
2023-01-01|100
2024-01-01|-110
] (delimiter is '|');
```

이는 첫 번째 예와 동일하지만 반대의 경우입니다. 여기서 우리는 1년 동안 100을 빌리고 10%의 이자로 갚습니다. 이 예에서 XIRR 계산은 첫 번째 예와 동일한 값인 0.1(10%)을 반환합니다.

첫 번째 예에서는 10의 이익을 얻었고 두 번째 예에서는 10의 손실을 경험했지만, 두 예에서 XIRR 기능의 반환 값은 양수입니다. 이는 XIRR 기능이 트랜잭션의 어느 쪽에 있든 관계없이 트랜잭션의 숨겨진 이자를 계산하기 때문입니다.

여러 솔루션의 제한 사항

Qlik의 XIRR 기능은 Rate 값이 해결되는 다음 방정식으로 정의됩니다.

```
XNPV(Rate, pmt, date) = 0
```

때때로 이 방정식에 하나 이상의 해가 있을 수 있습니다. 이는 "다중 IRR 문제"로 알려져 있으며 비정규 현금 흐름 스트림(비규칙적 현금 흐름이라고도 함)으로 인해 발생합니다. 다음 로드 스크립트는 이에 대한 예를 보여 줍니다.

```
Cashflow:
LOAD * inline [
Date|Payments
2021-01-01|-200
2022-01-01|500
2023-01-01|-250
] (delimiter is '|');
```

이 예에는 하나의 음성 솔루션과 하나의 양성 솔루션이 있습니다(Rate = -0.3 및 Rate = 0.8). **XIRR()**은 0.8을 반환합니다.

Qlik의 XIRR 기능은 솔루션을 검색할 때 Rate = 0에서 시작하여 솔루션을 찾을 때까지 단계적으로 속도를 높입니다. 둘 이상의 양성 솔루션이 있는 경우 가장 먼저 만나는 솔루션을 반환합니다. 양성 솔루션을 찾을 수 없으면 Rate를 다시 0으로 다시 설정하고 음성 방향에서 솔루션 검색을 시작합니다.

"정상적인" 현금 흐름 스트림에는 단 하나의 솔루션만 있음이 보장됩니다. "정상적인" 현금 흐름 스트림은 동일한 부호(양수 또는 음수)를 가진 모든 지불이 연속적인 그룹에 있음을 의미합니다.

관련 항목:

- *IRR - 차트 함수 (page 359)*
- *Aggr - 차트 함수 (page 524)*
- XIRR 함수 수정 및 업데이트

XNPV

XNPV()은 **pmt** 및 **date**에서 숫자 쌍으로 표현된 현금 흐름 일정(반드시 주기적일 필요는 없음)에 대해 집계된 순 현재 가치를 반환합니다. 모든 납입금은 1년 365일을 기준으로 할인됩니다.

구문:

```
XNPV(discount_rate, pmt, date)
```

반환 데이터 유형: 숫자

> *기본적으로 결과는 통화로 서식이 지정됩니다.*

XNPV를 계산하는 공식은 다음과 같습니다.

XNPV 집계 공식

$$XNPV = \sum_{i=1}^{n} \frac{P_i}{(1+rate)^{(di-d1)/365}}$$

설명:

- P_i = 단일 기간 i 동안의 순 현금 유입-유출
- d_1 = 첫 번째 지불 날짜
- d_i = i번째 지불 날짜
- rate = 할인율

순 현재 가치(NPV)는 할인율이 주어진 미래 현금 흐름 스트림의 현재 총 가치를 계산하는 데 사용됩니다. XNPV를 계산하려면 해당 날짜의 미래 현금 흐름을 추정해야 합니다. 이후 매 결제 시 결제일을 기준으로 복리할인율을 적용합니다.

일련의 지불에 대해 XNPV 집계를 수행하는 것은 해당 지불에 대해 Sum 집계를 수행하는 것과 유사합니다. 차이점은 각 금액이 선택한 할인율(이자율과 유사)과 지불이 얼마나 먼 미래에 있는지에 따라 수정(또는 "할인")된다는 것입니다. **discount_rate** 매개 변수를 0으로 설정한 상태에서 XNPV를 수행하면 XNPV가 Sum 연산과 동일해집니다(지불액은 합계되기 전에 수정되지 않음). 일반적으로 **discount_rate**가 0에 가까울수록 XNPV 결과는 Sum 집계의 결과와 유사합니다.

인수

인수	설명
discount_rate	**discount_rate**는 결제 금액을 할인해야 하는 연간 할인율입니다. 값이 0.1이면 10% 할인율을 나타냅니다.
pmt	지불. **date**에 제공된 지불 일정에 해당하는 현금 흐름을 포함하는 표현식 또는 필드. 양수 값은 유입으로 가정하고 음수 값은 유출로 가정합니다. ***XNPV()**는 항상 시작 날짜에 발생하므로 초기 현금 흐름을 할인하지 않습니다. 후속 지불은 1년 365일을 기준으로 할인됩니다. 이는 첫 번째 결제도 할인되는 **NPV()**와 다릅니다.*
date	**pmt**에 지정된 현금 흐름 지급에 해당하는 날짜의 일정을 포함하는 표현식 또는 필드입니다. 첫 번째 값은 미래 현금 흐름에 대한 시간 오프셋을 계산하기 위한 시작 날짜로 사용됩니다.

이 함수를 사용할 때 다음 제한 사항이 적용됩니다.

- 데이터 쌍의 한쪽 또는 양쪽에 텍스트 값, NULL 값, 누락된 값이 있으면 전체 데이터 쌍이 무시됩니다.

사용 시기

- XNPV()는 투자 기회의 순 현재 가치(NPV)를 계산하기 위한 재무 모델링에 사용됩니다.
- 모든 유형의 재무 모델의 경우 XNPV는 정밀도가 더 높으므로 NPV보다 선호됩니다.

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

예 1 – 단일 지불(스크립트)

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- CashFlow라는 테이블에 있는 한 프로젝트의 데이터 집합 및 1년 동안의 현금 흐름. 계산의 초기 날짜는 2022년 7월 1일로 설정되고 순 현금 흐름은 0입니다. 1년 후 $1000의 현금 흐름이 발생합니다.
- XNPV라는 테이블에서 프로젝트의 XNPV 필드를 계산하는 데 사용되는 CashFlow 테이블의 Resident LOAD.
- XNPV 계산에 사용되는 하드 코딩된 할인율 10%(0.1).
- Group By 문은 프로젝트에 대한 모든 지불을 그룹화하는 데 사용됩니다.

로드 스크립트

```
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;
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- PrjId
- XNPV

결과 테이블

PrjId	XNPV
1	$909.09

공식에 따르면 첫 번째 레코드의 XNPV 값은 0이고 두 번째 레코드의 XNPV 값은 $909.09이므로 총 XNPV는 $909.09입니다.

예 2 – 여러 지불(스크립트)

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- CashFlow라는 테이블에 있는 한 프로젝트의 데이터 집합 및 1년 동안의 현금 흐름.
- XNPV라는 테이블에서 프로젝트의 XNPV 필드를 계산하는 데 사용되는 CashFlow 테이블의 Resident LOAD.
- XNPV 계산에 사용되는 하드 코딩된 할인율 10%(0.1).
- Group By 문은 프로젝트에 대한 모든 지불을 그룹화하는 데 사용됩니다.

로드 스크립트

```
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;
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- PrjId
- XNPV

결과 테이블

PrjId	XNPV
1	$1322.21

이 예에서는 첫 해의 끝에 $1000의 지불을 받고 두 번째 해의 끝에 $500의 지불을 받습니다. 기간당 할인율이 10%인 유효 XNPV는 $1322.21입니다.

데이터의 첫 번째 행만 계산의 기준 날짜를 참조해야 합니다. 나머지 행의 경우 날짜 매개 변수가 경과 기간을 계산하는 데 사용되므로 순서는 중요하지 않습니다.

예 3 - 다중 지불 및 불규칙한 현금 흐름(스크립트)

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Project라는 테이블의 두 프로젝트에 대한 할인율.
- 프로젝트 ID 및 날짜별로 각 프로젝트에 대한 여러 기간의 현금 흐름. Dates 필드는 현금 흐름에 할인율이 적용되는 기간을 계산하는 데 사용됩니다. 첫 번째 레코드(초기 현금 흐름 및 날짜)를 제외하고 레코드의 순서는 중요하지 않으며 변경해도 계산에 영향을 미치지 않습니다.
- NoConcatenate, Resident LOAD 및 Left Join 함수를 조합하여 사용하면 Project 및 CashFlow 테이블의 레코드를 하나의 플랫 테이블로 결합한 임시 테이블 tmpNPV가 만들어집니다. 이 테이블에는 각 현금 흐름에 대해 반복되는 할인율이 있습니다.
- XNPV라는 테이블의 각 프로젝트에 대한 XNPV 필드를 계산하는 데 사용되는 tmpNPV 테이블의 Resident LOAD.
- only() 함수를 사용하여 각 프로젝트에 연결된 단일 값 할인율을 가져와 각 프로젝트에 대한 XNPV 계산에 사용합니다.
- 프로젝트 ID별로 그룹화하는 Group By 문은 각 프로젝트에 대한 모든 지불 및 해당 날짜를 그룹화하는 데 사용됩니다.
- 합성 데이터나 중복 데이터가 데이터 모델에 로드되는 것을 방지하기 위해 tmpXNPV 테이블은 스크립트 끝에서 삭제됩니다.

로드 스크립트

```
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;
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- PrjId
- XNPV

결과 테이블

PrjId	XNPV
1	$1735.54
2	$278.36

프로젝트 ID 1의 초기 현금 흐름은 2021년 7월 1일에 $0입니다. 2년 후 끝에 기간당 10%의 할인율로 $1000의 두 번 지불이 있습니다. 따라서 유효 XNPV는 $1735.54입니다.

프로젝트 ID 2는 2020년 7월 1일에 $1000(따라서 음수 부호)의 초기 유출이 있습니다. 2년 후 $500의 지불이 예상됩니다. 3년 후에는 $500의 추가 지불이 예상됩니다. 마지막으로 2024년 7월 1일에 $1000의 지불이 예상됩니다. 할인율이 15%인 유효 XNPV는 $278.36입니다.

관련 항목:

- *Drop table (page 147)*
- *group by (page 157)*
- *Join (page 70)*
- *Max (page 324)*
- *NoConcatenate (page 88)*
- *NPV - 차트 함수 (page 367)*
- *Only (page 334)*

XNPV - 차트 함수

XNPV()는 차트 차원에서 반복되는 **pmt** 및 **date**를 통해 지정된 표현식의 숫자 쌍으로 표현된 현금 흐름 일정(주기적일 필요는 없음)에 대해 집계된 순 현재 가치를 반환합니다. 모든 납입금은 1년 365일을 기준으로 할인됩니다.

구문:

```
XNPV([TOTAL [<fld{,fld}>]] discount_rate, pmt, date)
```

반환 데이터 유형: 숫자

> *기본적으로 결과는 통화로 서식이 지정됩니다.*

XNPV를 계산하는 공식은 다음과 같습니다.

XNPV 집계 공식

$$XNPV = \sum_{i=1}^{n} \frac{P_i}{(1+rate)^{(di-d1)/365}}$$

설명:

- P_i = 단일 기간 i 동안의 순 현금 유입-유출
- d_1 = 첫 번째 지불 날짜
- d_i = i번째 지불 날짜
- rate = 할인율

순 현재 가치(NPV)는 할인율이 주어진 미래 현금 흐름 스트림의 현재 총 가치를 계산하는 데 사용됩니다. XNPV를 계산하려면 해당 날짜의 미래 현금 흐름을 추정해야 합니다. 이후 매 결제 시 결제일을 기준으로 복리할인율을 적용합니다.

일련의 지불에 대해 XNPV 집계를 수행하는 것은 해당 지불에 대해 Sum 집계를 수행하는 것과 유사합니다. 차이점은 각 금액이 선택한 할인율(이자율과 유사)과 지불이 얼마나 먼 미래에 있는지에 따라 수정(또는 "할인")된다는 것입니다. **discount_rate** 매개 변수를 0으로 설정한 상태에서 XNPV를 수행하면 XNPV가 Sum 연산과 동일해집니다(지불액은 합계되기 전에 수정되지 않음). 일반적으로 **discount_rate**가 0에 가까울수록 XNPV 결과는 Sum 집계의 결과와 유사합니다.

인수

인수	설명
discount_rate	**discount_rate**는 결제 금액을 할인해야 하는 연간 할인율입니다. 값이 0.1이면 10% 할인율을 나타냅니다.
pmt	지불. **date**에 제공된 지불 일정에 해당하는 현금 흐름을 포함하는 표현식 또는 필드. 양수 값은 유입으로 가정하고 음수 값은 유출로 가정합니다. ***XNPV()**는 항상 시작 날짜에 발생하므로 초기 현금 흐름을 할인하지 않습니다. 후속 지불은 1년 365일을 기준으로 할인됩니다. 이는 첫 번째 결제도 할인되는 **NPV()**와 다릅니다.*
date	**pmt**에 지정된 현금 흐름 지급에 해당하는 날짜의 일정을 포함하는 표현식 또는 필드입니다. 첫 번째 값은 미래 현금 흐름에 대한 시간 오프셋을 계산하기 위한 시작 날짜로 사용됩니다.
TOTAL	**TOTAL**이 함수 인수 앞에 오는 경우, 현재 선택을 고려하되 현재 차원 값에 관련되지 않은 가능한 모든 값에 대한 계산이 실행됩니다. 즉, 차트 차원은 무시됩니다. **TOTAL [<fld {.fld}>]**(여기서 **TOTAL** 한정자 뒤에는 하나 이상의 필드 이름 목록이 차트 차원 변수의 하위 집합으로 옴)을 사용하여 가능한 전체 값의 하위 집합을 만듭니다.

이 함수를 사용할 때 다음 제한 사항이 적용됩니다.

- 내부 집계에 **TOTAL** 또는 **ALL** 한정자가 포함된 경우를 제외하고, **discount_rate**, **pmt** 및 **date**에 집계 함수가 포함되지 않아야 합니다. 고급 중첩 집계가 필요한 경우는 고급 함수 **Aggr**을 지정된 차원과 함께 사용하십시오.
- 데이터 쌍의 한쪽 또는 양쪽에 텍스트 값, NULL 값, 누락된 값이 있으면 전체 데이터 쌍이 무시됩니다.

사용 시기

- XNPV()는 투자 기회의 순 현재 가치(NPV)를 계산하기 위한 재무 모델링에 사용됩니다.
- 모든 유형의 재무 모델의 경우 XNPV는 정밀도가 더 높으므로 NPV보다 선호됩니다.

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 `SET DateFormat` 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

예

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 현금 흐름 트랜잭션이 포함된 데이터 집합.
- Cashflow라는 테이블에 저장된 정보.

로드 스크립트

```
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 '|');
```

결과

다음과 같이 하십시오.

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 계산을 측정값으로 추가합니다.

```
=XNPV(0.09, Payments, Date)
```

결과 테이블

=XNPV(0.09, Payments, Date)
$3062.49

관련 항목:

- *NPV - 차트 함수 (page 367)*
- *Aggr - 차트 함수 (page 524)*

통계 집계 함수

각 함수는 개요가 끝난 후에 더 자세히 설명합니다. 구문에서 함수 이름을 클릭하여 해당 함수에 대한 상세 설명에 즉시 액세스할 수도 있습니다.

데이터 로드 스크립트의 통계 집계 함수

다음 통계 집계 함수를 스크립트에서 사용할 수 있습니다.

Avg

Avg()는 **group by** 절로 정의된 여러 레코드에서 표현식의 집계된 데이터의 평균 값을 찾습니다.

```
Avg  ([distinct] expression)
```

Correl

Correl()은 **group by** 절로 정의된 여러 레코드에서 반복되는 x-expression 및 y-expression의 숫자 쌍으로 표현된 일련의 좌표에 대해 집계된 상관 계수를 반환합니다.

```
Correl (x-expression, y-expression)
```

Fractile

Fractile()은 **group by** 절로 정의된 여러 레코드에서 표현식의 집계된 데이터의 포괄 분위수(변위치)에 해당하는 값을 찾습니다.

```
Fractile (expression, fractile)
```

FractileExc

FractileExc()은 **group by** 절로 정의된 여러 레코드에서 표현식의 집계된 데이터의 단독 분위수(변위치)에 해당하는 값을 찾습니다.

```
FractileExc (expression, fractile)
```

Kurtosis

Kurtosis()는 **group by** 절로 정의된 여러 레코드에서 표현식의 데이터에 대한 첨도를 반환합니다.

```
Kurtosis  ([distinct ] expression )
```

LINEST_B

LINEST_B()는 **group by** 절로 정의된 여러 레코드에서 반복되는 x-expression 및 y-expression의 숫자 쌍으로 표현된 일련의 좌표에 대한 등식 y=mx+b로 정의된 선형 회귀의 집계된 b 값(y 절편)을 반환합니다.

```
LINEST_B (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_df

LINEST_DF()는 **group by** 절로 정의된 여러 레코드에서 반복되는 x-expression 및 y-expression의 숫자 쌍으로 표현된 일련의 좌표에 대한 등식 y=mx+b로 정의된 선형 회귀의 집계된 자유도를 반환합니다.

```
LINEST_DF  (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_f
이 스크립트 함수는 **group by** 절로 정의된 여러 레코드에서 반복되는 x-expression 및 y-expression의 숫자 쌍으로 표현된 일련의 좌표에 대한 등식 y=mx+b로 정의된 선형 회귀의 집계된 F 통계($r^2/(1-r^2)$)를 반환합니다.

```
LINEST_F (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_m
LINEST_M()은 **group by** 절로 정의된 여러 레코드에서 반복되는 x-expression 및 y-expression의 숫자 쌍으로 표현된 일련의 좌표에 대한 등식 y=mx+b로 정의된 선형 회귀의 집계된 m 값(경사)을 반환합니다.

```
LINEST_M  (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_r2
LINEST_R2()는 **group by** 절로 정의된 여러 레코드에서 반복되는 x-expression 및 y-expression의 숫자 쌍으로 표현된 일련의 좌표에 대한 등식 y=mx+b로 정의된 선형 회귀의 집계된 r^2 값(결정 계수)을 반환합니다.

```
LINEST_R2 (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_seb
LINEST_SEB()은 **group by** 절로 정의된 여러 레코드에서 반복되는 x-expression 및 y-expression의 숫자 쌍으로 표현된 일련의 좌표에 대한 등식 y=mx+b로 정의된 선형 회귀의 b 값의 집계된 표준 오차를 반환합니다.

```
LINEST_SEB  (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_sem
LINEST_SEM()은 **group by** 절로 정의된 여러 레코드에서 반복되는 x-expression 및 y-expression의 숫자 쌍으로 표현된 일련의 좌표에 대한 등식 y=mx+b로 정의된 선형 회귀의 m 값의 집계된 표준 오차를 반환합니다.

```
LINEST_SEM  (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_sey
LINEST_SEY()는 **group by** 절로 정의된 여러 레코드에서 반복되는 x-expression 및 y-expression의 숫자 쌍으로 표현된 일련의 좌표에 대한 등식 y=mx+b로 정의된 선형 회귀의 y 어림값에 대해 집계된 표준 오차를 반환합니다.

```
LINEST_SEY  (y-expression, x-expression [, y0 [, x0 ]])
```

LINEST_ssreg
LINEST_SSREG()는 **group by** 절로 정의된 여러 레코드에서 반복되는 x-expression 및 y-expression의 숫자 쌍으로 표현된 일련의 좌표에 대한 등식 y=mx+b로 정의된 선형 회귀의 집계된 회귀 제곱합을 반환합니다.

```
LINEST_SSREG  (y-expression, x-expression [, y0 [, x0 ]])
```

Linest_ssresid
LINEST_SSRESID()는 **group by** 절로 정의된 여러 레코드에서 반복되는 x-expression 및 y-expression의 숫자 쌍으로 표현된 일련의 좌표에 대한 등식 y=mx+b로 정의된 선형 회귀의 집계된 잔차 제곱합을 반환합니다.

```
LINEST_SSRESID  (y-expression, x-expression [, y0 [, x0 ]])
```

Median

Median()은 **group by** 절로 정의된 여러 레코드에서 표현식의 값에 대해 집계된 중앙값을 반환합니다.

```
Median (expression)
```

Skew

Skew()는 **group by** 절로 정의된 여러 레코드에서 표현식의 왜곡도를 반환합니다.

```
Skew ([ distinct] expression)
```

Stdev

Stdev()는 **group by** 절로 정의된 여러 레코드에서 표현식으로 지정한 값의 표준 편차를 반환합니다.

```
Stdev ([distinct] expression)
```

Sterr

Sterr()은 **group by** 절로 정의된 여러 레코드에서 반복되는 표현식으로 표현된 일련의 값에 대해 집계된 표준 오차(stdev/sqrt(n))를 반환합니다.

```
Sterr  ([distinct] expression)
```

STEYX

STEYX()는 **group by** 절로 정의된 여러 레코드에서 반복되는 x-expression 및 y-expression의 숫자 쌍으로 표현된 일련의 좌표에 대한 회귀 내의 각 x 값에 대해 예측된 y 값의 집계된 표준 오차를 반환합니다.

```
STEYX  (y-expression, x-expression)
```

차트 표현식의 통계 집계 함수

다음 통계 집계 함수를 차트에서 사용할 수 있습니다.

Avg

Avg()는 차트 차원에서 반복되는 표현식 또는 필드의 집계된 평균을 반환합니다.

```
Avg - 차트 함수({[SetExpression] [DISTINCT] [TOTAL [<fld{, fld}>]]} expr)
```

Correl

Correl()은 두 데이터 셋에 대해 집계된 상관 계수를 반환합니다. 상관 함수는 데이터 셋 간의 관계에 대한 측정값으로, 차트 차원에서 반복되는 (x,y) 값 쌍에 대해 집계됩니다.

```
Correl - 차트 함수({[SetExpression] [TOTAL [<fld {, fld}>]]} value1, value2 )
```

Fractile

Fractile() 은 차트 차원에서 반복되는 표현식을 통해 지정된 범위에서 집계된 데이터의 포괄 분위수(사분위수)에 해당하는 값을 찾습니다.

```
Fractile - 차트 함수({[SetExpression] [TOTAL [<fld {, fld}>]]} expr, fraction)
```

FractileExc

FractileExc() 은 차트 차원에서 반복되는 표현식을 통해 지정된 범위에서 집계된 데이터의 단독 분위수(사분위수)에 해당하는 값을 찾습니다.

FractileExc - 차트 함수({[SetExpression] [**TOTAL** [<fld {, fld}>]]} expr, fraction**)**

Kurtosis

Kurtosis()는 차트 차원에서 반복되는 표현식 또는 필드에서 집계된 데이터 범위의 첨도를 찾습니다.

Kurtosis - 차트 함수({[SetExpression] [**DISTINCT**] [**TOTAL** [<fld{, fld}>]]} expr**)**

LINEST_b

LINEST_B()는 차트 차원에서 반복되는 표현식 **x_value** 및 **y_value**를 통해 지정된 표현식의 숫자 쌍으로 표현된 일련의 좌표에 대한 등식 y=mx+b로 정의된 선형 회귀의 집계된 b 값(y 절편)을 반환합니다.

LINEST_R2 - 차트 함수({[SetExpression] [**TOTAL** [<fld{ ,fld}>]] }y_value, x_value [, y0_const[, x0_const]]**)**

LINEST_df

LINEST_DF()는 차트 차원에서 반복되는 **x_value** 및 **y_value**를 통해 지정된 표현식의 숫자 쌍으로 표현된 일련의 좌표에 대한 등식 y=mx+b로 정의된 선형 회귀의 집계된 자유도를 반환합니다.

LINEST_DF - 차트 함수({[SetExpression] [TOTAL [<fld{, fld}>]]} y_value, x_value [, y0_const [, x0_const]]**)**

LINEST_f

LINEST_F()는 차트 차원에서 반복되는 **x_value** 및 **y_value**를 통해 지정된 표현식의 숫자 쌍으로 표현된 일련의 좌표에 대한 등식 y=mx+b로 정의된 선형 회귀의 집계된 F 통계 (r2/(1-r2))를 반환합니다.

LINEST_F - 차트 함수({[SetExpression] [**TOTAL**[<fld{, fld}>]]} y_value, x_value [, y0_const [, x0_const]]**)**

LINEST_m

LINEST_M()은 차트 차원에서 반복되는 표현식 **x_value** 및 **y_value**를 통해 지정된 숫자 쌍으로 표현된 일련의 좌표에 대한 등식 y=mx+b로 정의된 선형 회귀의 집계된 m 값(경사)을 반환합니다.

LINEST_M - 차트 함수({[SetExpression] [**TOTAL**[<fld{, fld}>]]} y_value, x_value [, y0_const [, x0_const]]**)**

LINEST_r2

LINEST_R2()는 차트 차원에서 반복되는 표현식 **x_value** 및 **y_value**를 통해 지정된 숫자 쌍으로 표현된 일련의 좌표에 대한 등식 y=mx+b로 정의된 선형 회귀의 집계된 r2 값(결정 계수)을 반환합니다.

LINEST_R2 - 차트 함수({[SetExpression] [**TOTAL** [<fld{ ,fld}>]] }y_value, x_value [, y0_const[, x0_const]]**)**

LINEST_seb

LINEST_SEB()은 차트 차원에서 반복되는 표현식 **x_value** 및 **y_value**를 통해 지정된 숫자 쌍으로 표현된 일련의 좌표에 대한 등식 y=mx+b로 정의된 선형 회귀의 b 값의 집계된 표준 오차를 반환합니다.

LINEST_SEB - 차트 함수({[SetExpression] [**TOTAL** [<fld{ ,fld}>]] }y_value, x_value[, y0_const[, x0_const]]**)**

LINEST_sem

LINEST_SEM()은 차트 차원에서 반복되는 표현식 **x_value** 및 **y_value**를 통해 지정된 숫자 쌍으로 표현된 일련의 좌표에 대한 등식 y=mx+b로 정의된 선형 회귀의 m 값의 집계된 표준 오차를 반환합니다.

LINEST_SEM - 차트 함수([{set_expression}][**distinct**] [**total** [<fld {,fld}>]] y-expression, x-expression [, y0 [, x0]] **)**

LINEST_sey

LINEST_SEY()는 차트 차원에서 반복되는 표현식 **x_value** 및 **y_value**를 통해 지정된 숫자 쌍으로 표현된 일련의 좌표에 대한 등식 y=mx+b로 정의된 선형 회귀의 y 어림값의 집계된 표준 오차를 반환합니다.

LINEST_SEY - 차트 함수({[SetExpression] [**TOTAL** [<fld{ ,fld}>]] }y_value, x_value[, y0_const[, x0_const]]**)**

LINEST_ssreg

LINEST_SSREG()는 차트 차원에서 반복되는 표현식 **x_value** 및 **y_value**를 통해 지정된 숫자 쌍으로 표현된 일련의 좌표에 대한 등식 y=mx+b로 정의된 선형 회귀의 집계된 회귀 제곱합을 반환합니다.

LINEST_SSREG - 차트 함수({[SetExpression] [**TOTAL** [<fld{ ,fld}>]] }y_value, x_value[, y0_const[, x0_const]]**)**

LINEST_ssresid

LINEST_SSRESID()는 차트 차원에서 반복되는 **x_value** 및 **y_value**를 통해 지정된 표현식의 숫자 쌍으로 표현된 일련의 좌표에 대한 등식 y=mx+b로 정의된 선형 회귀의 집계된 잔차 제곱합을 반환합니다.

LINEST_SSRESID - 차트 함수LINEST_SSRESID()는 차트 차원에서 반복되는 x_value 및 y_value를 통해 지정된 표현식의 숫자 쌍으로 표현된 일련의 좌표에 대한 등식 y=mx+b로 정의된 선형 회귀의 집계된 잔차 제곱합을 반환합니다. LINEST_SSRESID([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value[, y0_const[, x0_const]]) 숫자 인수인수설명y_valuey 값 범위를 포함하는 표현식 또는 필드가 측정됩니다.x_valuex 값 범위를 포함하는 표현식 또는 필드가 측정됩니다.y0, x0회귀선이 일정한 지점에서 y 축을 통과하도록 선택적 값 y0을 지정할 수 있습니다. y0 및 x0을 모두 지정하면 회귀선이 하나의 고정된 좌표를 통과하도록 만들 수 있습니다. y0 및 x0을 모두 지정하지 않은 경우 이 함수를 계산하려면 유효한 데이터 쌍이 최소 2개 이상 필요합니다. y0 및 x0을 지정했다면 하나의 데이터 쌍으로 계산이 가능합니다. SetExpression기본적으로 집계 함수는 선택에 의해 정의된 사용 가능한 레코드의 집합을 집계합니다. 집합 분석 표현식으로 대체 레코드 집합을 정의할 수 있습니다. DISTINCT함수 인수 앞에 DISTINCT라는 단어가 있을 경우 해당 함수 인수의 평가 결과로 생성된 중복이 무시됩니다. TOTALTOTAL이 함수 인수 앞에 오는 경우, 현재 선택을 고려하되 현재 차원 값에 관련되지 않은 가능한 모든 값에 대한 계산이 실행됩니다. 즉, 차트 차원은 무시됩니다. TOTAL [<fld {.fld}>](여기서 TOTAL 한정자 뒤에는 하나 이상의 필드 이름 목록이 차트 차원 변수의 하위 집합으로 옴)을 사용하여 가능한 전체 값의 하위 집합을 만듭니다.회귀선이 일정한 지점에서 y 축을 통과하도록 선택적 값 y0을 지정할 수 있습니다. y0 및 x0을 모두 지정하면 회귀선이 하나의 고정된 좌표를 통과하도록 만들 수 있습니다. 집계 함수의 매개 변수는 이러한 내부 집계에 TOTAL 한정자가 포함되어 있지 않는 한 다른 집계 함수를 포함하지 않아야 합니다. 고급 중첩 집계가 필요한 경우는 고급 함수 Aggr을 지정된 차원과 함께 사용하십시오. 데이터 쌍의 한쪽 또는 양쪽에 텍스트 값, NULL 값, 누락된 값이 있으면 전체 데이터 쌍이 무시됩니다. An example of how to use linest functionsavg({[SetExpression] [**TOTAL** [<fld{ ,fld}>]] }y_value, x_value[, y0_const[, x0_const]]**)**

Median

Median()은 차트 차원에서 반복되는 표현식에서 집계된 값 범위의 평균 값을 반환합니다.

```
Median - 차트 함수({[SetExpression] [TOTAL [<fld{, fld}>]]} expr)
```

MutualInfo

MutualInfo는 두 필드 사이 또는 **Aggr()**에서 집계된 값 사이의 MI(상호 정보)를 계산합니다.

```
MutualInfo - 차트 함수{[SetExpression] [DISTINCT] [TOTAL target, driver [,
datatype [, breakdownbyvalue [, samplesize ]]])
```

Skew

Skew()는 차트 차원에서 반복되는 표현식 또는 필드의 집계된 왜곡도를 반환합니다.

```
Skew - 차트 함수{[SetExpression] [DISTINCT] [TOTAL [<fld{ ,fld}>]]} expr)
```

Stdev

Stdev()는 차트 차원에서 반복되는 표현식 또는 필드에서 집계된 데이터 범위의 표준 편차를 찾습니다.

```
Stdev - 차트 함수({[SetExpression] [DISTINCT] [TOTAL [<fld{, fld}>]]} expr)
```

Sterr

Sterr()는 차트 차원에서 반복되는 표현식에서 집계된 일련의 값에 대한 평균값의 표준 오차 값 (stdev/sqrt (n))을 찾습니다.

```
Sterr - 차트 함수({[SetExpression] [DISTINCT] [TOTAL[<fld{, fld}>]]} expr)
```

STEYX

STEYX()는 **y_value** 및 **x_value**를 통해 지정된 표현식의 숫자 쌍으로 표현된 일련의 좌표에 의해 지정된 선형 회귀의 각 x 값에 대해 y 값을 예측하는 경우의 집계된 표준 오차를 반환합니다.

```
STEYX - 차트 함수{[SetExpression] [TOTAL [<fld{, fld}>]]} y_value, x_value)
```

Avg

Avg()는 **group by** 절로 정의된 여러 레코드에서 표현식의 집계된 데이터의 평균 값을 찾습니다.

구문:

```
Avg([DISTINCT] expr)
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
expr	측정할 데이터가 포함된 표현식 또는 필드입니다.
DISTINCT	표현식 앞에 **distinct**라는 단어가 있을 경우 모든 중복 항목이 무시됩니다.

예 및 결과:

예제 스크립트를 앱에 추가하고 실행합니다. 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.

결과 데이터

예	결과
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 다음 측정값이 포함된 테이블을 만들어 시트에서 확인할 수 있습니다. Sum(Sales)/12
이전 예에서처럼 **Temp** 테이블이 로드된 것으로 가정합니다. 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 고유 값만 계수됩니다. 합계를 중복되지 않은 값의 수로 나눕니다.

Avg - 차트 함수

Avg()는 차트 차원에서 반복되는 표현식 또는 필드의 집계된 평균을 반환합니다.

구문:

```
Avg([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr)
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
expr	측정할 데이터가 포함된 표현식 또는 필드입니다.
SetExpression	기본적으로 집계 함수는 선택에 의해 정의된 사용 가능한 레코드의 집합을 집계합니다. 집합 분석 표현식으로 대체 레코드 집합을 정의할 수 있습니다.
DISTINCT	함수 인수 앞에 **DISTINCT**라는 단어가 있을 경우 해당 함수 인수의 평가 결과로 생성된 중복이 무시됩니다.
TOTAL	**TOTAL**이 함수 인수 앞에 오는 경우, 현재 선택을 고려하되 현재 차원 값에 관련되지 않은 가능한 모든 값에 대한 계산이 실행됩니다. 즉, 차트 차원은 무시됩니다. **TOTAL [<fld {.fld}>]**(여기서 **TOTAL** 한정자 뒤에는 하나 이상의 필드 이름 목록이 차트 차원 변수의 하위 집합으로 옴)을 사용하여 가능한 전체 값의 하위 집합을 만듭니다.

제한 사항:

집계 함수의 매개 변수는 이러한 내부 집계에 **TOTAL** 한정자가 포함되어 있지 않는 한 다른 집계 함수를 포함하지 않아야 합니다. 고급 중첩 집계가 필요한 경우는 고급 함수 **Aggr**을 지정된 차원과 함께 사용하십시오.

예 및 결과:

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

함수 예

예	결과
`Avg(Sales)`	차원 `Customer`와 측정값 `Avg([Sales])`가 포함된 테이블의 경우, **합계**가 표시된다면 결과는 2566입니다.
`Avg([TOTAL (Sales))`	`Customer`의 모든 값에 대해 53.458333이며, TOTAL 한정자는 차원이 무시됨을 의미하기 때문입니다.
`Avg(DISTINCT (Sales))`	합계에 대해 51.862069이며, Distinct 한정자를 사용하는 것은 각 `Customer`의 `Sales`에서 고유한 값만이 평가됨을 의미하기 때문입니다.

예에서 사용된 데이터:

```
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 '|');
```

관련 항목:

Aggr - 차트 함수 (page 524)

Correl

Correl()은 **group by** 절로 정의된 여러 레코드에서 반복되는 x-expression 및 y-expression의 숫자 쌍으로 표현된 일련의 좌표에 대해 집계된 상관 계수를 반환합니다.

구문:

```
Correl(value1, value2)
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
value1, value2	상관 계수가 측정될 두 샘플 집합이 포함된 표현식 또는 필드.

제한 사항:

데이터 쌍의 한쪽 또는 양쪽에 텍스트 값, NULL 값, 누락된 값이 있으면 전체 데이터 쌍이 무시됩니다.

예 및 결과:

예제 스크립트를 앱에 추가하고 실행합니다. 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.

결과 데이터

예	결과
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;	차원 Correl_Salary가 있는 테이블에서 데이터 로드 스크립트의 Correl() 계산 결과는 0.9270611로 표시됩니다.

Correl - 차트 함수

Correl()은 두 데이터 셋에 대해 집계된 상관 계수를 반환합니다. 상관 함수는 데이터 셋 간의 관계에 대한 측정값으로, 차트 차원에서 반복되는 (x,y) 값 쌍에 대해 집계됩니다.

구문:

```
Correl([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] value1, value2 )
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
value1, value2	상관 계수가 측정될 두 샘플 집합이 포함된 표현식 또는 필드.
SetExpression	기본적으로 집계 함수는 선택에 의해 정의된 사용 가능한 레코드의 집합을 집계합니다. 집합 분석 표현식으로 대체 레코드 집합을 정의할 수 있습니다.
DISTINCT	함수 인수 앞에 **DISTINCT**라는 단어가 있을 경우 해당 함수 인수의 평가 결과로 생성된 중복이 무시됩니다.
TOTAL	**TOTAL**이 함수 인수 앞에 오는 경우, 현재 선택을 고려하되 현재 차원 값에 관련되지 않은 가능한 모든 값에 대한 계산이 실행됩니다. 즉, 차트 차원은 무시됩니다. **TOTAL [<fld {.fld}>]**(여기서 **TOTAL** 한정자 뒤에는 하나 이상의 필드 이름 목록이 차트 차원 변수의 하위 집합으로 옴)을 사용하여 가능한 전체 값의 하위 집합을 만듭니다.

제한 사항:

집계 함수의 매개 변수는 이러한 내부 집계에 **TOTAL** 한정자가 포함되어 있지 않는 한 다른 집계 함수를 포함하지 않아야 합니다. 고급 중첩 집계가 필요한 경우는 고급 함수 **Aggr**을 지정된 차원과 함께 사용하십시오.

데이터 쌍의 한쪽 또는 양쪽에 텍스트 값, NULL 값, 누락된 값이 있으면 전체 데이터 쌍이 무시됩니다.

예 및 결과:

함수 예

예	결과
`Correl(Age, Salary)`	차원 `Employee name`와 측정값 `Correl(Age, Salary)`가 포함된 테이블의 경우, 결과가 0.9270611입니다. 결과는 합계 셀에만 표시됩니다.
`Correl(TOTAL Age, Salary))`	0.927. 이 결과와 다음 결과는 가독성을 위해 소수점 세 자리까지 표시됩니다. Gender 차원이 있는 필터 창을 만들고 선택 작업을 수행하면 Female을 선택한 경우 0.951이, Male을 선택한 경우 0.939가 결과로 표시됩니다. 이는 Gender의 다른 값에 속하지 않는 모든 결과가 선택에서 제외되기 때문입니다.
`Correl({1} TOTAL Age, Salary))`	0.927. 선택과는 관계가 없습니다. 집합 표현식 {1}이 모든 선택 및 차원을 무시하기 때문입니다.
`Correl(TOTAL <Gender> Age, Salary))`	합계 셀은 0.927, Male의 모든 값에 대해서는 0.939, Female의 모든 값에 대해서는 0.951입니다. 이는 Gender에 기반한 필터 창에서 선택을 수행한 결과에 해당합니다.

예에서 사용된 데이터:

```
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 '|');
```

관련 항목:

- *Aggr - 차트 함수 (page 524)*
- *Avg - 차트 함수 (page 389)*
- *RangeCorrel (page 1287)*

Fractile

Fractile()은 **group by** 절로 정의된 여러 레코드에서 표현식의 집계된 데이터의 포괄 분위수(변위치)에 해당하는 값을 찾습니다.

FractileExc (page 397)를 사용하여 단독 분위수를 계산할 수 있습니다.

구문:

```
Fractile(expr, fraction)
```

반환 데이터 유형: 숫자

이 함수는 `rank = fraction * (N-1) + 1`로 정의된 순위에 해당하는 값을 반환합니다. 여기서 `N`은 `expr`의 값 수입니다. `rank`가 정수가 아닌 경우 가장 가까운 두 값 사이에 보간이 이루어집니다.

인수:

인수

인수	설명
expr	분위수를 계산할 때 사용할 데이터가 포함된 표현식 또는 필드입니다.
fraction	분위수에 대응하는 0 ~ 1 사이의 숫자(분위수로 표현된 사분위수)가 계산됩니다.

예 및 결과:

예제 스크립트를 앱에 추가하고 실행합니다. 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.

결과 데이터

예	결과
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;	차원 Type 및 MyFractile가 있는 테이블에서 데이터 로드 스크립트의 Fractile() 계산 결과는 다음과 같습니다. Type MyFractile Comparison 27.5 Observation 36

Fractile - 차트 함수

Fractile() 은 차트 차원에서 반복되는 표현식을 통해 지정된 범위에서 집계된 데이터의 포괄 분위수(사분위수)에 해당하는 값을 찾습니다.

FractileExc - 차트 함수 (page 399)를 사용하여 단독 분위수를 계산할 수 있습니다.

구문:

```
Fractile([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr, fraction)
```

반환 데이터 유형: 숫자

이 함수는 `rank = fraction * (N-1) + 1`로 정의된 순위에 해당하는 값을 반환합니다. 여기서 `N`은 `expr`의 값 수입니다. `rank`가 정수가 아닌 경우 가장 가까운 두 값 사이에 보간이 이루어집니다.

인수:

인수

인수	설명
expr	분위수를 계산할 때 사용할 데이터가 포함된 표현식 또는 필드입니다.
fraction	분위수에 대응하는 0 ~ 1 사이의 숫자(분위수로 표현된 사분위수)가 계산됩니다.
SetExpression	기본적으로 집계 함수는 선택에 의해 정의된 사용 가능한 레코드의 집합을 집계합니다. 집합 분석 표현식으로 대체 레코드 집합을 정의할 수 있습니다.
DISTINCT	함수 인수 앞에 **DISTINCT**라는 단어가 있을 경우 해당 함수 인수의 평가 결과로 생성된 중복이 무시됩니다.
TOTAL	**TOTAL**이 함수 인수 앞에 오는 경우, 현재 선택을 고려하되 현재 차원 값에 관련되지 않은 가능한 모든 값에 대한 계산이 실행됩니다. 즉, 차트 차원은 무시됩니다. **TOTAL [<fld {.fld}>]**(여기서 **TOTAL** 한정자 뒤에는 하나 이상의 필드 이름 목록이 차트 차원 변수의 하위 집합으로 옴)을 사용하여 가능한 전체 값의 하위 집합을 만듭니다.

제한 사항:

집계 함수의 매개 변수는 이러한 내부 집계에 **TOTAL** 한정자가 포함되어 있지 않는 한 다른 집계 함수를 포함하지 않아야 합니다. 고급 중첩 집계가 필요한 경우는 고급 함수 **Aggr**을 지정된 차원과 함께 사용하십시오.

예 및 결과:

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

함수 예

예	결과
Fractile (Sales, 0.75)	차원 Customer와 측정값 Fractile([Sales])가 포함된 테이블의 경우, **합계**가 표시된다면 결과는 71.75입니다. 이는 Sales 값의 75%가 속하는 값의 분포 지점입니다.
Fractile(TOTAL Sales, 0.75))	Customer의 모든 값에 대해 71.75이며, TOTAL 한정자는 차원이 무시됨을 의미하기 때문입니다.
Fractile (DISTINCT Sales, 0.75)	합계에 대해 70이며, DISTINCT 한정자를 사용하는 것은 각 Customer의 Sales에서 고유한 값만이 평가됨을 의미하기 때문입니다.

예에서 사용된 데이터:

```
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 '|');
```

관련 항목:

Aggr - 차트 함수 (page 524)

FractileExc

FractileExc()은 **group by** 절로 정의된 여러 레코드에서 표현식의 집계된 데이터의 단독 분위수(변위치)에 해당하는 값을 찾습니다.

Fractile (page 394)를 사용하여 포괄 분위수를 계산할 수 있습니다.

구문:

```
FractileExc(expr, fraction)
```

반환 데이터 유형: 숫자

이 함수는 rank = fraction * (N+1)로 정의된 순위에 해당하는 값을 반환합니다. 여기서 N은 expr의 값 수입니다. rank가 정수가 아닌 경우 가장 가까운 두 값 사이에 보간이 이루어집니다.

인수:

인수

인수	설명
expr	분위수를 계산할 때 사용할 데이터가 포함된 표현식 또는 필드입니다.
fraction	분위수에 대응하는 0 ~ 1 사이의 숫자(분위수로 표현된 사분위수)가 계산됩니다.

예 및 결과:

예제 스크립트를 앱에 추가하고 실행합니다. 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.

결과 데이터

예	결과
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;	차원 Type 및 MyFractile가 있는 테이블에서 데이터 로드 스크립트의 FractileExc() 계산 결과는 다음과 같습니다. Type MyFractile Comparison 28.5 Observation 38

FractileExc - 차트 함수

FractileExc() 은 차트 차원에서 반복되는 표현식을 통해 지정된 범위에서 집계된 데이터의 단독 분위수(사분위수)에 해당하는 값을 찾습니다.

Fractile - 차트 함수 (page 395)를 사용하여 포괄 분위수를 계산할 수 있습니다.

구문:

```
FractileExc([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr,
fraction)
```

반환 데이터 유형: 숫자

이 함수는 rank = fraction * (N+1)로 정의된 순위에 해당하는 값을 반환합니다. 여기서 N은 expr의 값 수입니다. rank가 정수가 아닌 경우 가장 가까운 두 값 사이에 보간이 이루어집니다.

인수:

인수

인수	설명
expr	분위수를 계산할 때 사용할 데이터가 포함된 표현식 또는 필드입니다.
fraction	분위수에 대응하는 0 ~ 1 사이의 숫자(분위수로 표현된 사분위수)가 계산됩니다.
SetExpression	기본적으로 집계 함수는 선택에 의해 정의된 사용 가능한 레코드의 집합을 집계합니다. 집합 분석 표현식으로 대체 레코드 집합을 정의할 수 있습니다.
DISTINCT	함수 인수 앞에 **DISTINCT**라는 단어가 있을 경우 해당 함수 인수의 평가 결과로 생성된 중복이 무시됩니다.
TOTAL	**TOTAL**이 함수 인수 앞에 오는 경우, 현재 선택을 고려하되 현재 차원 값에 관련되지 않은 가능한 모든 값에 대한 계산이 실행됩니다. 즉, 차트 차원은 무시됩니다. **TOTAL [<fld {.fld}>]**(여기서 **TOTAL** 한정자 뒤에는 하나 이상의 필드 이름 목록이 차트 차원 변수의 하위 집합으로 옴)을 사용하여 가능한 전체 값의 하위 집합을 만듭니다.

제한 사항:

집계 함수의 매개 변수는 이러한 내부 집계에 **TOTAL** 한정자가 포함되어 있지 않는 한 다른 집계 함수를 포함하지 않아야 합니다. 고급 중첩 집계가 필요한 경우는 고급 함수 **Aggr**을 지정된 차원과 함께 사용하십시오.

예 및 결과:

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

함수 예

예	결과
`FractileExc (Sales, 0.75)`	차원 `Customer`와 측정값 `FractileExc([Sales])`가 포함된 테이블의 경우, **합계**가 표시된다면 결과는 75.25입니다. 이는 `Sales` 값의 75%가 속하는 값의 분포 지점입니다.
`FractileExc (TOTAL Sales, 0.75))`	`Customer`의 모든 값에 대해 75.25이며, TOTAL 한정자는 차원이 무시됨을 의미하기 때문입니다.
`FractileExc (DISTINCT Sales, 0.75)`	합계에 대해 73.50이며, DISTINCT 한정자를 사용하는 것은 각 `Customer`의 `Sales`에서 고유한 값만이 평가됨을 의미하기 때문입니다.

예에서 사용된 데이터:

```
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 '|');
```

관련 항목:

Aggr - 차트 함수 (page 524)

Kurtosis

Kurtosis()는 **group by** 절로 정의된 여러 레코드에서 표현식의 데이터에 대한 첨도를 반환합니다.

구문:

```
Kurtosis([distinct ] expr )
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
expr	측정할 데이터가 포함된 표현식 또는 필드입니다.
distinct	표현식 앞에 **distinct**라는 단어가 있을 경우 모든 중복 항목이 무시됩니다.

예 및 결과:

예제 스크립트를 앱에 추가하고 실행합니다. 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.

결과 데이터

예	결과
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;	차원 Type, MyKurtosis1 및 MyKurtosis2가 있는 테이블에서 데이터 로드 스크립트의 Kurtosis() 계산 결과는 다음과 같습니다. Type MyKurtosis1 MyKurtosis2 Comparison -1.1612957 -1.4982366 Observation -1.1148768 -0.93540144

Kurtosis - 차트 함수

Kurtosis()는 차트 차원에서 반복되는 표현식 또는 필드에서 집계된 데이터 범위의 첨도를 찾습니다.

구문:

```
Kurtosis([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr)
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
expr	측정할 데이터가 포함된 표현식 또는 필드입니다.
SetExpression	기본적으로 집계 함수는 선택에 의해 정의된 사용 가능한 레코드의 집합을 집계합니다. 집합 분석 표현식으로 대체 레코드 집합을 정의할 수 있습니다.
DISTINCT	함수 인수 앞에 **DISTINCT**라는 단어가 있을 경우 해당 함수 인수의 평가 결과로 생성된 중복이 무시됩니다.
TOTAL	**TOTAL**이 함수 인수 앞에 오는 경우, 현재 선택을 고려하되 현재 차원 값에 관련되지 않은 가능한 모든 값에 대한 계산이 실행됩니다. 즉, 차트 차원은 무시됩니다. **TOTAL [<fld {.fld}>]**(여기서 **TOTAL** 한정자 뒤에는 하나 이상의 필드 이름 목록이 차트 차원 변수의 하위 집합으로 옴)을 사용하여 가능한 전체 값의 하위 집합을 만듭니다.

제한 사항:

집계 함수의 매개 변수는 이러한 내부 집계에 **TOTAL** 한정자가 포함되어 있지 않는 한 다른 집계 함수를 포함하지 않아야 합니다. 고급 중첩 집계가 필요한 경우는 고급 함수 **Aggr**을 지정된 차원과 함께 사용하십시오.

예 및 결과:

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

함수 예

예	결과
`Kurtosis (Value)`	차원 `Type` 및 측정값 `Kurtosis(Value)`가 포함된 테이블의 경우, 테이블에 **합계**가 표시되고 숫자 서식이 유효 숫자 3개로 설정되었다면 결과는 1.252입니다. `Comparison`의 경우 1.161이며 `Observation`의 경우 1.115입니다.
`Kurtosis (TOTAL Value))`	`Type`의 모든 값에 대해 1.252이며, TOTAL 한정자는 차원이 무시됨을 의미하기 때문입니다.

예에서 사용된 데이터:

```
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 '|');
```

관련 항목:

Avg - 차트 함수 (page 389)

LINEST_B

LINEST_B()는 **group by** 절로 정의된 여러 레코드에서 반복되는 x-expression 및 y-expression의 숫자 쌍으로 표현된 일련의 좌표에 대한 등식 y=mx+b로 정의된 선형 회귀의 집계된 b 값(y 절편)을 반환합니다.

구문:

```
LINEST_B (y_value, x_value[, y0 [, x0 ]])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
y_value	y 값 범위를 포함하는 표현식 또는 필드가 측정됩니다.
x_value	x 값 범위를 포함하는 표현식 또는 필드가 측정됩니다.

인수	설명
y(0), x(0)	회귀선이 일정한 지점에서 y 축을 통과하도록 선택적 값 y0을 지정할 수 있습니다. y0 및 x0을 모두 지정하면 회귀선이 하나의 고정된 좌표를 통과하도록 만들 수 있습니다. y0 및 x0을 모두 지정하지 않은 경우 이 함수를 계산하려면 유효한 데이터 쌍이 최소 2개 이상 필요합니다. y0 및 x0을 지정했다면 하나의 데이터 쌍으로 계산이 가능합니다.

제한 사항:

데이터 쌍의 한쪽 또는 양쪽에 텍스트 값, NULL 값, 누락된 값이 있으면 전체 데이터 쌍이 무시됩니다.

관련 항목:

linest 함수를 사용하는 방법의 예 (page 443)

LINEST_B - 차트 함수

LINEST_B()는 차트 차원에서 반복되는 표현식 **x_value** 및 **y_value**를 통해 지정된 표현식의 숫자 쌍으로 표현된 일련의 좌표에 대한 등식 y=mx+b로 정의된 선형 회귀의 집계된 b 값(y 절편)을 반환합니다.

구문:

```
LINEST_B([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value
[, y0_const [ , x0_const]])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
y_value	y 값 범위를 포함하는 표현식 또는 필드가 측정됩니다.
x_value	x 값 범위를 포함하는 표현식 또는 필드가 측정됩니다.
y0_const, x0_const	회귀선이 일정한 지점에서 y 축을 통과하도록 선택적 값 y0을 지정할 수 있습니다. y0 및 x0을 모두 지정하면 회귀선이 하나의 고정된 좌표를 통과하도록 만들 수 있습니다. *y0 및 x0을 모두 지정하지 않은 경우 이 함수를 계산하려면 유효한 데이터 쌍이 최소 2개 이상 필요합니다. y0 및 x0을 지정했다면 하나의 데이터 쌍으로 계산이 가능합니다.*
SetExpression	기본적으로 집계 함수는 선택에 의해 정의된 사용 가능한 레코드의 집합을 집계합니다. 집합 분석 표현식으로 대체 레코드 집합을 정의할 수 있습니다.
DISTINCT	함수 인수 앞에 **DISTINCT**라는 단어가 있을 경우 해당 함수 인수의 평가 결과로 생성된 중복이 무시됩니다.

인수	설명
TOTAL	**TOTAL**이 함수 인수 앞에 오는 경우, 현재 선택을 고려하되 현재 차원 값에 관련되지 않은 가능한 모든 값에 대한 계산이 실행됩니다. 즉, 차트 차원은 무시됩니다. **TOTAL [<fld {.fld}>]**(여기서 **TOTAL** 한정자 뒤에는 하나 이상의 필드 이름 목록이 차트 차원 변수의 하위 집합으로 옴)을 사용하여 가능한 전체 값의 하위 집합을 만듭니다.

제한 사항:

집계 함수의 매개 변수는 이러한 내부 집계에 **TOTAL** 한정자가 포함되어 있지 않는 한 다른 집계 함수를 포함하지 않아야 합니다. 고급 중첩 집계가 필요한 경우는 고급 함수 **Aggr**을 지정된 차원과 함께 사용하십시오.

데이터 쌍의 한쪽 또는 양쪽에 텍스트 값, NULL 값, 누락된 값이 있으면 전체 데이터 쌍이 무시됩니다.

관련 항목:

linest 함수를 사용하는 방법의 예 (page 443)
Avg - 차트 함수 (page 389)

LINEST_DF

LINEST_DF()는 **group by** 절로 정의된 여러 레코드에서 반복되는 x-expression 및 y-expression의 숫자 쌍으로 표현된 일련의 좌표에 대한 등식 y=mx+b로 정의된 선형 회귀의 집계된 자유도를 반환합니다.

구문:

```
LINEST_DF (y_value, x_value[, y0 [, x0 ]])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
y_value	y 값 범위를 포함하는 표현식 또는 필드가 측정됩니다.
x_value	x 값 범위를 포함하는 표현식 또는 필드가 측정됩니다.
y(0), x(0)	회귀선이 일정한 지점에서 y 축을 통과하도록 선택적 값 y0을 지정할 수 있습니다. y0 및 x0을 모두 지정하면 회귀선이 하나의 고정된 좌표를 통과하도록 만들 수 있습니다. y0 및 x0을 모두 지정하지 않은 경우 이 함수를 계산하려면 유효한 데이터 쌍이 최소 2개 이상 필요합니다. y0 및 x0을 지정했다면 하나의 데이터 쌍으로 계산이 가능합니다.

제한 사항:

데이터 쌍의 한쪽 또는 양쪽에 텍스트 값, NULL 값, 누락된 값이 있으면 전체 데이터 쌍이 무시됩니다.

관련 항목:

- *linest 함수를 사용하는 방법의 예 (page 443)*

LINEST_DF - 차트 함수

LINEST_DF()는 차트 차원에서 반복되는 **x_value** 및 **y_value**를 통해 지정된 표현식의 숫자 쌍으로 표현된 일련의 좌표에 대한 등식 y=mx+b로 정의된 선형 회귀의 집계된 자유도를 반환합니다.

구문:

```
LINEST_DF([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value [, y0_const [, x0_const]])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
y_value	y 값 범위를 포함하는 표현식 또는 필드가 측정됩니다.
x_value	x 값 범위를 포함하는 표현식 또는 필드가 측정됩니다.
y0, x0	회귀선이 일정한 지점에서 y 축을 통과하도록 선택적 값 y0을 지정할 수 있습니다. y0 및 x0을 모두 지정하면 회귀선이 하나의 고정된 좌표를 통과하도록 만들 수 있습니다. *y0 및 x0을 모두 지정하지 않은 경우 이 함수를 계산하려면 유효한 데이터 쌍이 최소 2개 이상 필요합니다. y0 및 x0을 지정했다면 하나의 데이터 쌍으로 계산이 가능합니다.*
SetExpression	기본적으로 집계 함수는 선택에 의해 정의된 사용 가능한 레코드의 집합을 집계합니다. 집합 분석 표현식으로 대체 레코드 집합을 정의할 수 있습니다.
DISTINCT	함수 인수 앞에 **DISTINCT**라는 단어가 있을 경우 해당 함수 인수의 평가 결과로 생성된 중복이 무시됩니다.
TOTAL	**TOTAL**이 함수 인수 앞에 오는 경우, 현재 선택을 고려하되 현재 차원 값에 관련되지 않은 가능한 모든 값에 대한 계산이 실행됩니다. 즉, 차트 차원은 무시됩니다. **TOTAL [<fld {.fld}>]**(여기서 **TOTAL** 한정자 뒤에는 하나 이상의 필드 이름 목록이 차트 차원 변수의 하위 집합으로 옴)을 사용하여 가능한 전체 값의 하위 집합을 만듭니다.

제한 사항:

집계 함수의 매개 변수는 이러한 내부 집계에 **TOTAL** 한정자가 포함되어 있지 않는 한 다른 집계 함수를 포함하지 않아야 합니다. 고급 중첩 집계가 필요한 경우는 고급 함수 **Aggr**을 지정된 차원과 함께 사용하십시오.

데이터 쌍의 한쪽 또는 양쪽에 텍스트 값, NULL 값, 누락된 값이 있으면 전체 데이터 쌍이 무시됩니다.

관련 항목:

- *linest 함수를 사용하는 방법의 예 (page 443)*
- *Avg - 차트 함수 (page 389)*

LINEST_F

이 스크립트 함수는 **group by** 절로 정의된 여러 레코드에서 반복되는 x-expression 및 y-expression의 숫자 쌍으로 표현된 일련의 좌표에 대한 등식 y=mx+b로 정의된 선형 회귀의 집계된 F 통계($r^2/(1-r^2)$)를 반환합니다.

구문:

```
LINEST_F (y_value, x_value[, y0 [, x0 ]])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
y_value	y 값 범위를 포함하는 표현식 또는 필드가 측정됩니다.
x_value	x 값 범위를 포함하는 표현식 또는 필드가 측정됩니다.
y(0), x(0)	회귀선이 일정한 지점에서 y 축을 통과하도록 선택적 값 y0을 지정할 수 있습니다. y0 및 x0을 모두 지정하면 회귀선이 하나의 고정된 좌표를 통과하도록 만들 수 있습니다. y0 및 x0을 모두 지정하지 않은 경우 이 함수를 계산하려면 유효한 데이터 쌍이 최소 2개 이상 필요합니다. y0 및 x0을 지정했다면 하나의 데이터 쌍으로 계산이 가능합니다.

제한 사항:

데이터 쌍의 한쪽 또는 양쪽에 텍스트 값, NULL 값, 누락된 값이 있으면 전체 데이터 쌍이 무시됩니다.

관련 항목:

- *linest 함수를 사용하는 방법의 예 (page 443)*

LINEST_F - 차트 함수

LINEST_F()는 차트 차원에서 반복되는 **x_value** 및 **y_value**를 통해 지정된 표현식의 숫자 쌍으로 표현된 일련의 좌표에 대한 등식 y=mx+b로 정의된 선형 회귀의 집계된 F 통계 (r2/(1-r2))를 반환합니다.

구문:

```
LINEST_F([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value
[, y0_const [, x0_const]])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
y_value	y 값 범위를 포함하는 표현식 또는 필드가 측정됩니다.
x_value	x 값 범위를 포함하는 표현식 또는 필드가 측정됩니다.
y0, x0	회귀선이 일정한 지점에서 y 축을 통과하도록 선택적 값 y0을 지정할 수 있습니다. y0 및 x0을 모두 지정하면 회귀선이 하나의 고정된 좌표를 통과하도록 만들 수 있습니다. *y0 및 x0을 모두 지정하지 않은 경우 이 함수를 계산하려면 유효한 데이터 쌍이 최소 2개 이상 필요합니다. y0 및 x0을 지정했다면 하나의 데이터 쌍으로 계산이 가능합니다.*
SetExpression	기본적으로 집계 함수는 선택에 의해 정의된 사용 가능한 레코드의 집합을 집계합니다. 집합 분석 표현식으로 대체 레코드 집합을 정의할 수 있습니다.
DISTINCT	함수 인수 앞에 **DISTINCT**라는 단어가 있을 경우 해당 함수 인수의 평가 결과로 생성된 중복이 무시됩니다.
TOTAL	**TOTAL**이 함수 인수 앞에 오는 경우, 현재 선택을 고려하되 현재 차원 값에 관련되지 않은 가능한 모든 값에 대한 계산이 실행됩니다. 즉, 차트 차원은 무시됩니다. **TOTAL [<fld {.fld}>]**(여기서 **TOTAL** 한정자 뒤에는 하나 이상의 필드 이름 목록이 차트 차원 변수의 하위 집합으로 옴)을 사용하여 가능한 전체 값의 하위 집합을 만듭니다.

제한 사항:

집계 함수의 매개 변수는 이러한 내부 집계에 **TOTAL** 한정자가 포함되어 있지 않는 한 다른 집계 함수를 포함하지 않아야 합니다. 고급 중첩 집계가 필요한 경우는 고급 함수 **Aggr**을 지정된 차원과 함께 사용하십시오.

데이터 쌍의 한쪽 또는 양쪽에 텍스트 값, NULL 값, 누락된 값이 있으면 전체 데이터 쌍이 무시됩니다.

관련 항목:

- *linest 함수를 사용하는 방법의 예 (page 443)*
- *Avg - 차트 함수 (page 389)*

LINEST_M

LINEST_M()은 **group by** 절로 정의된 여러 레코드에서 반복되는 x-expression 및 y-expression의 숫자 쌍으로 표현된 일련의 좌표에 대한 등식 y=mx+b로 정의된 선형 회귀의 집계된 m 값(경사)을 반환합니다.

구문:

```
LINEST_M (y_value, x_value[, y0 [, x0 ]])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
y_value	y 값 범위를 포함하는 표현식 또는 필드가 측정됩니다.
x_value	x 값 범위를 포함하는 표현식 또는 필드가 측정됩니다.
y(0), x(0)	회귀선이 일정한 지점에서 y 축을 통과하도록 선택적 값 y0을 지정할 수 있습니다. y0 및 x0을 모두 지정하면 회귀선이 하나의 고정된 좌표를 통과하도록 만들 수 있습니다. y0 및 x0을 모두 지정하지 않은 경우 이 함수를 계산하려면 유효한 데이터 쌍이 최소 2개 이상 필요합니다. y0 및 x0을 지정했다면 하나의 데이터 쌍으로 계산이 가능합니다.

제한 사항:

데이터 쌍의 한쪽 또는 양쪽에 텍스트 값, NULL 값, 누락된 값이 있으면 전체 데이터 쌍이 무시됩니다.

관련 항목:

linest 함수를 사용하는 방법의 예 (page 443)

LINEST_M - 차트 함수

LINEST_M()은 차트 차원에서 반복되는 표현식 **x_value** 및 **y_value**를 통해 지정된 숫자 쌍으로 표현된 일련의 좌표에 대한 등식 y=mx+b로 정의된 선형 회귀의 집계된 m 값(경사)을 반환합니다.

구문:

```
LINEST_M([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value
[, y0_const [, x0_const]])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
y_value	y 값 범위를 포함하는 표현식 또는 필드가 측정됩니다.
x_value	x 값 범위를 포함하는 표현식 또는 필드가 측정됩니다.

인수	설명
y0, x0	회귀선이 일정한 지점에서 y 축을 통과하도록 선택적 값 y0을 지정할 수 있습니다. y0 및 x0을 모두 지정하면 회귀선이 하나의 고정된 좌표를 통과하도록 만들 수 있습니다. *y0 및 x0을 모두 지정하지 않은 경우 이 함수를 계산하려면 유효한 데이터 쌍이 최소 2개 이상 필요합니다. y0 및 x0을 지정했다면 하나의 데이터 쌍으로 계산이 가능합니다.*
SetExpression	기본적으로 집계 함수는 선택에 의해 정의된 사용 가능한 레코드의 집합을 집계합니다. 집합 분석 표현식으로 대체 레코드 집합을 정의할 수 있습니다.
DISTINCT	함수 인수 앞에 **DISTINCT**라는 단어가 있을 경우 해당 함수 인수의 평가 결과로 생성된 중복이 무시됩니다.
TOTAL	**TOTAL**이 함수 인수 앞에 오는 경우, 현재 선택을 고려하되 현재 차원 값에 관련되지 않은 가능한 모든 값에 대한 계산이 실행됩니다. 즉, 차트 차원은 무시됩니다. **TOTAL [<fld {.fld}>]**(여기서 **TOTAL** 한정자 뒤에는 하나 이상의 필드 이름 목록이 차트 차원 변수의 하위 집합으로 옴)을 사용하여 가능한 전체 값의 하위 집합을 만듭니다.

제한 사항:

집계 함수의 매개 변수는 이러한 내부 집계에 **TOTAL** 한정자가 포함되어 있지 않는 한 다른 집계 함수를 포함하지 않아야 합니다. 고급 중첩 집계가 필요한 경우는 고급 함수 **Aggr**을 지정된 차원과 함께 사용하십시오.

데이터 쌍의 한쪽 또는 양쪽에 텍스트 값, NULL 값, 누락된 값이 있으면 전체 데이터 쌍이 무시됩니다.

관련 항목:

- *linest 함수를 사용하는 방법의 예 (page 443)*
- *Avg - 차트 함수 (page 389)*

LINEST_R2

LINEST_R2()는 **group by** 절로 정의된 여러 레코드에서 반복되는 x-expression 및 y-expression의 숫자 쌍으로 표현된 일련의 좌표에 대한 등식 y=mx+b로 정의된 선형 회귀의 집계된 r^2 값(결정 계수)을 반환합니다.

구문:

```
LINEST_R2 (y_value, x_value[, y0 [, x0 ]])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
y_value	y 값 범위를 포함하는 표현식 또는 필드가 측정됩니다.
x_value	x 값 범위를 포함하는 표현식 또는 필드가 측정됩니다.
y(0), x(0)	회귀선이 일정한 지점에서 y 축을 통과하도록 선택적 값 y0을 지정할 수 있습니다. y0 및 x0을 모두 지정하면 회귀선이 하나의 고정된 좌표를 통과하도록 만들 수 있습니다. y0 및 x0을 모두 지정하지 않은 경우 이 함수를 계산하려면 유효한 데이터 쌍이 최소 2개 이상 필요합니다. y0 및 x0을 지정했다면 하나의 데이터 쌍으로 계산이 가능합니다.

제한 사항:

데이터 쌍의 한쪽 또는 양쪽에 텍스트 값, NULL 값, 누락된 값이 있으면 전체 데이터 쌍이 무시됩니다.

관련 항목:

linest 함수를 사용하는 방법의 예 (page 443)

LINEST_R2 - 차트 함수

LINEST_R2()는 차트 차원에서 반복되는 표현식 **x_value** 및 **y_value**를 통해 지정된 숫자 쌍으로 표현된 일련의 좌표에 대한 등식 y=mx+b로 정의된 선형 회귀의 집계된 r2 값(결정 계수)을 반환합니다.

구문:

```
LINEST_R2([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value[, y0_const[, x0_const]])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
y_value	y 값 범위를 포함하는 표현식 또는 필드가 측정됩니다.
x_value	x 값 범위를 포함하는 표현식 또는 필드가 측정됩니다.

인수	설명
y0, x0	회귀선이 일정한 지점에서 y 축을 통과하도록 선택적 값 y0을 지정할 수 있습니다. y0 및 x0을 모두 지정하면 회귀선이 하나의 고정된 좌표를 통과하도록 만들 수 있습니다. *y0 및 x0을 모두 지정하지 않은 경우 이 함수를 계산하려면 유효한 데이터 쌍이 최소 2개 이상 필요합니다. y0 및 x0을 지정했다면 하나의 데이터 쌍으로 계산이 가능합니다.*
SetExpression	기본적으로 집계 함수는 선택에 의해 정의된 사용 가능한 레코드의 집합을 집계합니다. 집합 분석 표현식으로 대체 레코드 집합을 정의할 수 있습니다.
DISTINCT	함수 인수 앞에 **DISTINCT**라는 단어가 있을 경우 해당 함수 인수의 평가 결과로 생성된 중복이 무시됩니다.
TOTAL	**TOTAL**이 함수 인수 앞에 오는 경우, 현재 선택을 고려하되 현재 차원 값에 관련되지 않은 가능한 모든 값에 대한 계산이 실행됩니다. 즉, 차트 차원은 무시됩니다. **TOTAL [<fld {.fld}>]**(여기서 **TOTAL** 한정자 뒤에는 하나 이상의 필드 이름 목록이 차트 차원 변수의 하위 집합으로 옴)을 사용하여 가능한 전체 값의 하위 집합을 만듭니다.

제한 사항:

집계 함수의 매개 변수는 이러한 내부 집계에 **TOTAL** 한정자가 포함되어 있지 않는 한 다른 집계 함수를 포함하지 않아야 합니다. 고급 중첩 집계가 필요한 경우는 고급 함수 **Aggr**을 지정된 차원과 함께 사용하십시오.

데이터 쌍의 한쪽 또는 양쪽에 텍스트 값, NULL 값, 누락된 값이 있으면 전체 데이터 쌍이 무시됩니다.

관련 항목:

- *linest 함수를 사용하는 방법의 예 (page 443)*
- *Avg - 차트 함수 (page 389)*

LINEST_SEB

LINEST_SEB()은 **group by** 절로 정의된 여러 레코드에서 반복되는 x-expression 및 y-expression의 숫자 쌍으로 표현된 일련의 좌표에 대한 등식 y=mx+b로 정의된 선형 회귀의 b 값의 집계된 표준 오차를 반환합니다.

구문:

```
LINEST_SEB (y_value, x_value[, y0 [, x0 ]])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
y_value	y 값 범위를 포함하는 표현식 또는 필드가 측정됩니다.
x_value	x 값 범위를 포함하는 표현식 또는 필드가 측정됩니다.
y(0), x(0)	회귀선이 일정한 지점에서 y 축을 통과하도록 선택적 값 y0을 지정할 수 있습니다. y0 및 x0을 모두 지정하면 회귀선이 하나의 고정된 좌표를 통과하도록 만들 수 있습니다. y0 및 x0을 모두 지정하지 않은 경우 이 함수를 계산하려면 유효한 데이터 쌍이 최소 2개 이상 필요합니다. y0 및 x0을 지정했다면 하나의 데이터 쌍으로 계산이 가능합니다.

제한 사항:

데이터 쌍의 한쪽 또는 양쪽에 텍스트 값, NULL 값, 누락된 값이 있으면 전체 데이터 쌍이 무시됩니다.

관련 항목:

linest 함수를 사용하는 방법의 예 (page 443)

LINEST_SEB - 차트 함수

LINEST_SEB()은 차트 차원에서 반복되는 표현식 **x_value** 및 **y_value**를 통해 지정된 숫자 쌍으로 표현된 일련의 좌표에 대한 등식 y=mx+b로 정의된 선형 회귀의 b 값의 집계된 표준 오차를 반환합니다.

구문:

```
LINEST_SEB([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value[, y0_const[, x0_const]])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
y_value	y 값 범위를 포함하는 표현식 또는 필드가 측정됩니다.
x_value	x 값 범위를 포함하는 표현식 또는 필드가 측정됩니다.

인수	설명
y0, x0	회귀선이 일정한 지점에서 y 축을 통과하도록 선택적 값 y0을 지정할 수 있습니다. y0 및 x0을 모두 지정하면 회귀선이 하나의 고정된 좌표를 통과하도록 만들 수 있습니다. *y0 및 x0을 모두 지정하지 않은 경우 이 함수를 계산하려면 유효한 데이터 쌍이 최소 2개 이상 필요합니다. y0 및 x0을 지정했다면 하나의 데이터 쌍으로 계산이 가능합니다.*
SetExpression	기본적으로 집계 함수는 선택에 의해 정의된 사용 가능한 레코드의 집합을 집계합니다. 집합 분석 표현식으로 대체 레코드 집합을 정의할 수 있습니다.
DISTINCT	함수 인수 앞에 **DISTINCT**라는 단어가 있을 경우 해당 함수 인수의 평가 결과로 생성된 중복이 무시됩니다.
TOTAL	**TOTAL**이 함수 인수 앞에 오는 경우, 현재 선택을 고려하되 현재 차원 값에 관련되지 않은 가능한 모든 값에 대한 계산이 실행됩니다. 즉, 차트 차원은 무시됩니다. **TOTAL [<fld {.fld}>]**(여기서 **TOTAL** 한정자 뒤에는 하나 이상의 필드 이름 목록이 차트 차원 변수의 하위 집합으로 옴)을 사용하여 가능한 전체 값의 하위 집합을 만듭니다.

제한 사항:

집계 함수의 매개 변수는 이러한 내부 집계에 **TOTAL** 한정자가 포함되어 있지 않는 한 다른 집계 함수를 포함하지 않아야 합니다. 고급 중첩 집계가 필요한 경우는 고급 함수 **Aggr**을 지정된 차원과 함께 사용하십시오.

데이터 쌍의 한쪽 또는 양쪽에 텍스트 값, NULL 값, 누락된 값이 있으면 전체 데이터 쌍이 무시됩니다.

관련 항목:

- *linest 함수를 사용하는 방법의 예 (page 443)*
- *Avg - 차트 함수 (page 389)*

LINEST_SEM

LINEST_SEM()은 **group by** 절로 정의된 여러 레코드에서 반복되는 x-expression 및 y-expression의 숫자 쌍으로 표현된 일련의 좌표에 대한 등식 y=mx+b로 정의된 선형 회귀의 m 값의 집계된 표준 오차를 반환합니다.

구문:

```
LINEST_SEM (y_value, x_value[, y0 [, x0 ]])
```

반환 데이터 유형: 숫자

인수:

인수	설명
y_value	y 값 범위를 포함하는 표현식 또는 필드가 측정됩니다.
x_value	x 값 범위를 포함하는 표현식 또는 필드가 측정됩니다.
y(0), x(0)	회귀선이 일정한 지점에서 y 축을 통과하도록 선택적 값 y0을 지정할 수 있습니다. y0 및 x0을 모두 지정하면 회귀선이 하나의 고정된 좌표를 통과하도록 만들 수 있습니다. y0 및 x0을 모두 지정하지 않은 경우 이 함수를 계산하려면 유효한 데이터 쌍이 최소 2개 이상 필요합니다. y0 및 x0을 지정했다면 하나의 데이터 쌍으로 계산이 가능합니다.

제한 사항:

데이터 쌍의 한쪽 또는 양쪽에 텍스트 값, NULL 값, 누락된 값이 있으면 전체 데이터 쌍이 무시됩니다.

관련 항목:

linest 함수를 사용하는 방법의 예 (page 443)

LINEST_SEM - 차트 함수

LINEST_SEM()은 차트 차원에서 반복되는 표현식 **x_value** 및 **y_value**를 통해 지정된 숫자 쌍으로 표현된 일련의 좌표에 대한 등식 y=mx+b로 정의된 선형 회귀의 m 값의 집계된 표준 오차를 반환합니다.

구문:

```
LINEST_SEM([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value[, y0_const[, x0_const]])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
y_value	y 값 범위를 포함하는 표현식 또는 필드가 측정됩니다.
x_value	x 값 범위를 포함하는 표현식 또는 필드가 측정됩니다.

인수	설명
y0, x0	회귀선이 일정한 지점에서 y 축을 통과하도록 선택적 값 y0을 지정할 수 있습니다. y0 및 x0을 모두 지정하면 회귀선이 하나의 고정된 좌표를 통과하도록 만들 수 있습니다. *y0 및 x0을 모두 지정하지 않은 경우 이 함수를 계산하려면 유효한 데이터 쌍이 최소 2개 이상 필요합니다. y0 및 x0을 지정했다면 하나의 데이터 쌍으로 계산이 가능합니다.*
SetExpression	기본적으로 집계 함수는 선택에 의해 정의된 사용 가능한 레코드의 집합을 집계합니다. 집합 분석 표현식으로 대체 레코드 집합을 정의할 수 있습니다.
DISTINCT	함수 인수 앞에 **DISTINCT**라는 단어가 있을 경우 해당 함수 인수의 평가 결과로 생성된 중복이 무시됩니다.
TOTAL	**TOTAL**이 함수 인수 앞에 오는 경우, 현재 선택을 고려하되 현재 차원 값에 관련되지 않은 가능한 모든 값에 대한 계산이 실행됩니다. 즉, 차트 차원은 무시됩니다. **TOTAL [<fld {.fld}>]**(여기서 **TOTAL** 한정자 뒤에는 하나 이상의 필드 이름 목록이 차트 차원 변수의 하위 집합으로 옴)을 사용하여 가능한 전체 값의 하위 집합을 만듭니다.

제한 사항:

집계 함수의 매개 변수는 이러한 내부 집계에 **TOTAL** 한정자가 포함되어 있지 않는 한 다른 집계 함수를 포함하지 않아야 합니다. 고급 중첩 집계가 필요한 경우는 고급 함수 **Aggr**을 지정된 차원과 함께 사용하십시오.

데이터 쌍의 한쪽 또는 양쪽에 텍스트 값, NULL 값, 누락된 값이 있으면 전체 데이터 쌍이 무시됩니다.

관련 항목:

- *linest 함수를 사용하는 방법의 예 (page 443)*
- *Avg - 차트 함수 (page 389)*

LINEST_SEY

LINEST_SEY()는 **group by** 절로 정의된 여러 레코드에서 반복되는 x-expression 및 y-expression의 숫자 쌍으로 표현된 일련의 좌표에 대한 등식 y=mx+b로 정의된 선형 회귀의 y 어림값에 대해 집계된 표준 오차를 반환합니다.

구문:

```
LINEST_SEY (y_value, x_value[, y0 [, x0 ]])
```

반환 데이터 유형: 숫자

인수:

인수	설명
y_value	y 값 범위를 포함하는 표현식 또는 필드가 측정됩니다.
x_value	x 값 범위를 포함하는 표현식 또는 필드가 측정됩니다.
y(0), x(0)	회귀선이 일정한 지점에서 y 축을 통과하도록 선택적 값 y0을 지정할 수 있습니다. y0 및 x0을 모두 지정하면 회귀선이 하나의 고정된 좌표를 통과하도록 만들 수 있습니다. y0 및 x0을 모두 지정하지 않은 경우 이 함수를 계산하려면 유효한 데이터 쌍이 최소 2개 이상 필요합니다. y0 및 x0을 지정했다면 하나의 데이터 쌍으로 계산이 가능합니다.

제한 사항:

데이터 쌍의 한쪽 또는 양쪽에 텍스트 값, NULL 값, 누락된 값이 있으면 전체 데이터 쌍이 무시됩니다.

관련 항목:

linest 함수를 사용하는 방법의 예 (page 443)

LINEST_SEY - 차트 함수

LINEST_SEY()는 차트 차원에서 반복되는 표현식 **x_value** 및 **y_value**를 통해 지정된 숫자 쌍으로 표현된 일련의 좌표에 대한 등식 y=mx+b로 정의된 선형 회귀의 y 어림값의 집계된 표준 오차를 반환합니다.

구문:

```
LINEST_SEY([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value[, y0_const[, x0_const]])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
y_value	y 값 범위를 포함하는 표현식 또는 필드가 측정됩니다.
x_value	x 값 범위를 포함하는 표현식 또는 필드가 측정됩니다.

인수	설명
y0, x0	회귀선이 일정한 지점에서 y 축을 통과하도록 선택적 값 y0을 지정할 수 있습니다. y0 및 x0을 모두 지정하면 회귀선이 하나의 고정된 좌표를 통과하도록 만들 수 있습니다. *y0 및 x0을 모두 지정하지 않은 경우 이 함수를 계산하려면 유효한 데이터 쌍이 최소 2개 이상 필요합니다. y0 및 x0을 지정했다면 하나의 데이터 쌍으로 계산이 가능합니다.*
SetExpression	기본적으로 집계 함수는 선택에 의해 정의된 사용 가능한 레코드의 집합을 집계합니다. 집합 분석 표현식으로 대체 레코드 집합을 정의할 수 있습니다.
DISTINCT	함수 인수 앞에 **DISTINCT**라는 단어가 있을 경우 해당 함수 인수의 평가 결과로 생성된 중복이 무시됩니다.
TOTAL	**TOTAL**이 함수 인수 앞에 오는 경우, 현재 선택을 고려하되 현재 차원 값에 관련되지 않은 가능한 모든 값에 대한 계산이 실행됩니다. 즉, 차트 차원은 무시됩니다. **TOTAL [<fld {.fld}>]**(여기서 **TOTAL** 한정자 뒤에는 하나 이상의 필드 이름 목록이 차트 차원 변수의 하위 집합으로 옴)을 사용하여 가능한 전체 값의 하위 집합을 만듭니다.

제한 사항:

집계 함수의 매개 변수는 이러한 내부 집계에 **TOTAL** 한정자가 포함되어 있지 않는 한 다른 집계 함수를 포함하지 않아야 합니다. 고급 중첩 집계가 필요한 경우는 고급 함수 **Aggr**을 지정된 차원과 함께 사용하십시오.

데이터 쌍의 한쪽 또는 양쪽에 텍스트 값, NULL 값, 누락된 값이 있으면 전체 데이터 쌍이 무시됩니다.

관련 항목:

- *linest 함수를 사용하는 방법의 예 (page 443)*
- *Avg - 차트 함수 (page 389)*

LINEST_SSREG

LINEST_SSREG()는 **group by** 절로 정의된 여러 레코드에서 반복되는 x-expression 및 y-expression의 숫자 쌍으로 표현된 일련의 좌표에 대한 등식 y=mx+b로 정의된 선형 회귀의 집계된 회귀 제곱합을 반환합니다.

구문:

```
LINEST_SSREG (y_value, x_value[, y0 [, x0 ]])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
y_value	y 값 범위를 포함하는 표현식 또는 필드가 측정됩니다.
x_value	x 값 범위를 포함하는 표현식 또는 필드가 측정됩니다.
y(0), x(0)	회귀선이 일정한 지점에서 y 축을 통과하도록 선택적 값 y0을 지정할 수 있습니다. y0 및 x0을 모두 지정하면 회귀선이 하나의 고정된 좌표를 통과하도록 만들 수 있습니다. y0 및 x0을 모두 지정하지 않은 경우 이 함수를 계산하려면 유효한 데이터 쌍이 최소 2개 이상 필요합니다. y0 및 x0을 지정했다면 하나의 데이터 쌍으로 계산이 가능합니다.

제한 사항:

데이터 쌍의 한쪽 또는 양쪽에 텍스트 값, NULL 값, 누락된 값이 있으면 전체 데이터 쌍이 무시됩니다.

관련 항목:

linest 함수를 사용하는 방법의 예 (page 443)

LINEST_SSREG - 차트 함수

LINEST_SSREG()는 차트 차원에서 반복되는 표현식 **x_value** 및 **y_value**를 통해 지정된 숫자 쌍으로 표현된 일련의 좌표에 대한 등식 y=mx+b로 정의된 선형 회귀의 집계된 회귀 제곱합을 반환합니다.

구문:

```
LINEST_SSREG([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value[, y0_const[, x0_const]])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
y_value	y 값 범위를 포함하는 표현식 또는 필드가 측정됩니다.
x_value	x 값 범위를 포함하는 표현식 또는 필드가 측정됩니다.

인수	설명
y0, x0	회귀선이 일정한 지점에서 y 축을 통과하도록 선택적 값 y0을 지정할 수 있습니다. y0 및 x0을 모두 지정하면 회귀선이 하나의 고정된 좌표를 통과하도록 만들 수 있습니다. *y0 및 x0을 모두 지정하지 않은 경우 이 함수를 계산하려면 유효한 데이터 쌍이 최소 2개 이상 필요합니다. y0 및 x0을 지정했다면 하나의 데이터 쌍으로 계산이 가능합니다.*
SetExpression	기본적으로 집계 함수는 선택에 의해 정의된 사용 가능한 레코드의 집합을 집계합니다. 집합 분석 표현식으로 대체 레코드 집합을 정의할 수 있습니다.
DISTINCT	함수 인수 앞에 **DISTINCT**라는 단어가 있을 경우 해당 함수 인수의 평가 결과로 생성된 중복이 무시됩니다.
TOTAL	**TOTAL**이 함수 인수 앞에 오는 경우, 현재 선택을 고려하되 현재 차원 값에 관련되지 않은 가능한 모든 값에 대한 계산이 실행됩니다. 즉, 차트 차원은 무시됩니다. **TOTAL [<fld {.fld}>]**(여기서 **TOTAL** 한정자 뒤에는 하나 이상의 필드 이름 목록이 차트 차원 변수의 하위 집합으로 옴)을 사용하여 가능한 전체 값의 하위 집합을 만듭니다.

제한 사항:

집계 함수의 매개 변수는 이러한 내부 집계에 **TOTAL** 한정자가 포함되어 있지 않는 한 다른 집계 함수를 포함하지 않아야 합니다. 고급 중첩 집계가 필요한 경우는 고급 함수 **Aggr**을 지정된 차원과 함께 사용하십시오.

데이터 쌍의 한쪽 또는 양쪽에 텍스트 값, NULL 값, 누락된 값이 있으면 전체 데이터 쌍이 무시됩니다.

관련 항목:

- *linest 함수를 사용하는 방법의 예 (page 443)*
- *Avg - 차트 함수 (page 389)*

LINEST_SSRESID

LINEST_SSRESID()는 **group by** 절로 정의된 여러 레코드에서 반복되는 x-expression 및 y-expression의 숫자 쌍으로 표현된 일련의 좌표에 대한 등식 y=mx+b로 정의된 선형 회귀의 집계된 잔차 제곱합을 반환합니다.

구문:

```
LINEST_SSRESID (y_value, x_value[, y0 [, x0 ]])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
y_value	y 값 범위를 포함하는 표현식 또는 필드가 측정됩니다.
x_value	x 값 범위를 포함하는 표현식 또는 필드가 측정됩니다.
y(0), x(0)	회귀선이 일정한 지점에서 y 축을 통과하도록 선택적 값 y0을 지정할 수 있습니다. y0 및 x0을 모두 지정하면 회귀선이 하나의 고정된 좌표를 통과하도록 만들 수 있습니다. y0 및 x0을 모두 지정하지 않은 경우 이 함수를 계산하려면 유효한 데이터 쌍이 최소 2개 이상 필요합니다. y0 및 x0을 지정했다면 하나의 데이터 쌍으로 계산이 가능합니다.

제한 사항:

데이터 쌍의 한쪽 또는 양쪽에 텍스트 값, NULL 값, 누락된 값이 있으면 전체 데이터 쌍이 무시됩니다.

관련 항목:

linest 함수를 사용하는 방법의 예 (page 443)

LINEST_SSRESID - 차트 함수

LINEST_SSRESID()는 차트 차원에서 반복되는 **x_value** 및 **y_value**를 통해 지정된 표현식의 숫자 쌍으로 표현된 일련의 좌표에 대한 등식 y=mx+b로 정의된 선형 회귀의 집계된 잔차 제곱합을 반환합니다.

구문:

```
LINEST_SSRESID([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value,
x_value[, y0_const[, x0_const]])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
y_value	y 값 범위를 포함하는 표현식 또는 필드가 측정됩니다.
x_value	x 값 범위를 포함하는 표현식 또는 필드가 측정됩니다.

인수	설명
y0, x0	회귀선이 일정한 지점에서 y 축을 통과하도록 선택적 값 y0을 지정할 수 있습니다. y0 및 x0을 모두 지정하면 회귀선이 하나의 고정된 좌표를 통과하도록 만들 수 있습니다. *y0 및 x0을 모두 지정하지 않은 경우 이 함수를 계산하려면 유효한 데이터 쌍이 최소 2개 이상 필요합니다. y0 및 x0을 지정했다면 하나의 데이터 쌍으로 계산이 가능합니다.*
SetExpression	기본적으로 집계 함수는 선택에 의해 정의된 사용 가능한 레코드의 집합을 집계합니다. 집합 분석 표현식으로 대체 레코드 집합을 정의할 수 있습니다.
DISTINCT	함수 인수 앞에 **DISTINCT**라는 단어가 있을 경우 해당 함수 인수의 평가 결과로 생성된 중복이 무시됩니다.
TOTAL	**TOTAL**이 함수 인수 앞에 오는 경우, 현재 선택을 고려하되 현재 차원 값에 관련되지 않은 가능한 모든 값에 대한 계산이 실행됩니다. 즉, 차트 차원은 무시됩니다. **TOTAL [<fld {.fld}>]**(여기서 **TOTAL** 한정자 뒤에는 하나 이상의 필드 이름 목록이 차트 차원 변수의 하위 집합으로 옴)을 사용하여 가능한 전체 값의 하위 집합을 만듭니다.

회귀선이 일정한 지점에서 y 축을 통과하도록 선택적 값 y0을 지정할 수 있습니다. y0 및 x0을 모두 지정하면 회귀선이 하나의 고정된 좌표를 통과하도록 만들 수 있습니다.

제한 사항:

집계 함수의 매개 변수는 이러한 내부 집계에 **TOTAL** 한정자가 포함되어 있지 않는 한 다른 집계 함수를 포함하지 않아야 합니다. 고급 중첩 집계가 필요한 경우는 고급 함수 **Aggr**을 지정된 차원과 함께 사용하십시오.

데이터 쌍의 한쪽 또는 양쪽에 텍스트 값, NULL 값, 누락된 값이 있으면 전체 데이터 쌍이 무시됩니다.

관련 항목:

- *linest 함수를 사용하는 방법의 예 (page 443)*
- *Avg - 차트 함수 (page 389)*

Median

Median()은 **group by** 절로 정의된 여러 레코드에서 표현식의 값에 대해 집계된 중앙값을 반환합니다.

구문:

```
Median (expr)
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
expr	측정할 데이터가 포함된 표현식 또는 필드입니다.

예: 중앙값을 사용한 스크립트 표현식

예 - 스크립트 표현식

로드 스크립트

이 예의 데이터 로드 편집기에서 다음 인라인 데이터 및 스크립트 표현식을 로드합니다.

```
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;
```

시각화 만들기

Letter 및 **MyMedian**을 차원으로 사용하여 Qlik Sense 시트에 테이블 시각화를 만듭니다.

결과

Letter	MyMedian
A	3.5
B	8

설명

중앙값은 숫자가 가장 작은 것부터 큰 것 순으로 정렬되었을 때 "중간" 숫자로 간주됩니다. 데이터 집합에 짝수 개의 값이 있는 경우 함수는 두 중간 값의 평균을 반환합니다. 이 예에서 중앙값은 각각 3.5 및 8인 **A** 및 **B** 값 집합에 대해 계산됩니다.

Median - 차트 함수

Median()은 차트 차원에서 반복되는 표현식에서 집계된 값 범위의 평균 값을 반환합니다.

구문:

```
Median([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr)
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
expr	측정할 데이터가 포함된 표현식 또는 필드입니다.
SetExpression	기본적으로 집계 함수는 선택에 의해 정의된 사용 가능한 레코드의 집합을 집계합니다. 집합 분석 표현식으로 대체 레코드 집합을 정의할 수 있습니다.
DISTINCT	함수 인수 앞에 **DISTINCT**라는 단어가 있을 경우 해당 함수 인수의 평가 결과로 생성된 중복이 무시됩니다.
TOTAL	**TOTAL**이 함수 인수 앞에 오는 경우, 현재 선택을 고려하되 현재 차원 값에 관련되지 않은 가능한 모든 값에 대한 계산이 실행됩니다. 즉, 차트 차원은 무시됩니다. **TOTAL [<fld {.fld}>]**(여기서 **TOTAL** 한정자 뒤에는 하나 이상의 필드 이름 목록이 차트 차원 변수의 하위 집합으로 옴)을 사용하여 가능한 전체 값의 하위 집합을 만듭니다.

제한 사항:

집계 함수의 매개 변수는 이러한 내부 집계에 **TOTAL** 한정자가 포함되어 있지 않는 한 다른 집계 함수를 포함하지 않아야 합니다. 고급 중첩 집계가 필요한 경우는 고급 함수 **Aggr**을 지정된 차원과 함께 사용하십시오.

예: 중앙값을 사용한 차트 표현식

예 - 차트 표현식

로드 스크립트

데이터 로드 편집기에서 다음 데이터를 인라인 로드로 로드하여 아래 차트 표현식 예를 만듭니다.

```
Load RecNo() as RowNo, Letter, Number Inline
[Letter, Number
A,1
A,3
A,4
A,9
B,2
B,8
B,9];
```

시각화 만들기

Letter를 차원으로 사용하여 Qlik Sense 시트에 테이블 시각화를 만듭니다.

차트 표현식

테이블에 다음 표현식을 측정값으로 추가합니다.

```
Median(Number)
```

결과

Letter	Median(Number)
Totals	4
A	3.5
B	8

설명

중앙값은 숫자가 가장 작은 것부터 큰 것 순으로 정렬되었을 때 "중간" 숫자로 간주됩니다. 데이터 집합에 짝수 개의 값이 있는 경우 함수는 두 중간 값의 평균을 반환합니다. 이 예에서 중앙값은 각각 3.5 및 8인 **A** 및 **B** 값 집합에 대해 계산됩니다.

합계의 중앙값은 모든 값에서 계산됩니다. 이는 4와 같습니다.

관련 항목:

Avg - 차트 함수 (page 389)

MutualInfo - 차트 함수

MutualInfo는 두 필드 사이 또는 **Aggr()**에서 집계된 값 사이의 MI(상호 정보)를 계산합니다.

MutualInfo는 두 데이터 집합에 대해 집계된 상호 정보를 반환합니다. 이를 통해 필드와 잠재적인 동인 간의 주요 동인 분석이 가능합니다. 상호 정보는 데이터 집합 간의 관계를 측정하고 차트 차원에서 반복되는 (x, y) 쌍 값에 대해 집계됩니다. 상호 정보는 0과 1 사이에서 측정되며 백분위 수 값으로 형식을 지정할 수 있습니다. **MutualInfo**는 선택 또는 집합 표현식에 의해 정의됩니다.

MutualInfo를 사용하면 다양한 종류의 MI 분석이 가능합니다.

- 쌍 단위 MI: 동인 필드와 대상 필드 사이의 MI를 계산합니다.
- 값으로 동인 분석: MI는 동인 필드 및 대상 필드의 개별 필드 값 사이에서 계산됩니다.
- 기능 선택: 그리드형 차트에서 **MutualInfo**를 사용하여 MI를 기반으로 모든 필드가 서로 비교되는 행렬을 만듭니다.

MutualInfo는 상호 정보를 공유하는 필드 간의 인과 관계를 반드시 나타내는 것은 아닙니다. 두 필드는 상호 정보를 공유할 수 있지만 서로 동일한 동인이 아닐 수 있습니다. 예를 들어, 아이스크림 판매량과 실외 온도를 비교할 때 **MutualInfo**는 둘 사이의 상호 정보를 표시합니다. 실외 온도가 아이스크림 판매를 유도하는지 여부(가능성 있음) 또는 아이스크림 판매가 실외 온도를 유도하는지 여부(가능성 낮음)는 표시하지 않습니다.

상호 정보를 계산할 때, 연결은 서로 다른 테이블에 있는 필드 값 간의 대응성과 및 빈도에 영향을 줍니다.

동일한 필드 또는 선택에 대해 반환되는 값은 약간 다를 수 있습니다. 이는 각 **MutualInfo** 호출이 무작위로 선택된 샘플과 **MutualInfo** 알고리즘에 내재하는 무작위성을 기반으로 작동하기 때문입니다.

MutualInfo는 **Aggr()** 함수에 적용할 수 있습니다.

구문:

```
MutualInfo({SetExpression}] [DISTINCT] [TOTAL] field1, field2 , datatype [, breakdownbyvalue [, samplesize ]])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
field1, field2	상호 정보가 측정될 두 샘플 집합이 포함된 표현식 또는 필드입니다.
datatype	대상 및 동인에 포함된 데이터 유형은 다음과 같습니다. 불연속:불연속인 경우, 1 또는 'dd' 연속:연속인 경우, 2 또는 'cc' 연속:불연속인 경우, 3 또는 'cd' 불연속:연속인 경우, 4 또는 'dc' 데이터 유형은 대/소문자를 구분하지 않습니다.
breakdownbyvalue	동인의 값에 해당하는 정적 값입니다. 제공된 경우 계산은 해당 값에 대한 MI 기여도를 계산합니다. **ValueList()** 또는 **ValueLoop()**를 사용할 수 있습니다. **Null()**이 추가되면 계산은 동인의 모든 값에 대한 전체 MI를 계산합니다. 값별로 분류하려면 동인에 불연속 데이터가 있어야 합니다.
samplesize	대상 및 동인에서 샘플링할 값의 개수입니다. 임의로 샘플링합니다. **MutualInfo**에는 최소 샘플 크기 80이 필요합니다. 기본적으로 **MutualInfo**는 리소스 중심일 수 있으므로 **MutualInfo**로 최대 10,000개 데이터 쌍만 샘플링합니다. 샘플 크기에서 더 많은 데이터 쌍을 지정할 수 있습니다. **MutualInfo** 시간이 초과되면 샘플 크기를 줄입니다.
SetExpression	기본적으로 집계 함수는 선택에 의해 정의된 사용 가능한 레코드의 집합을 집계합니다. 집합 분석 표현식으로 대체 레코드 집합을 정의할 수 있습니다.
DISTINCT	함수 인수 앞에 **DISTINCT**라는 단어가 있을 경우 해당 함수 인수의 평가 결과로 생성된 중복이 무시됩니다.
TOTAL	**TOTAL**이 함수 인수 앞에 오는 경우, 현재 선택을 고려하되 현재 차원 값에 관련되지 않은 가능한 모든 값에 대한 계산이 실행됩니다. 즉, 차트 차원은 무시됩니다. **TOTAL [<fld {.fld}>]**(여기서 **TOTAL** 한정자 뒤에는 하나 이상의 필드 이름 목록이 차트 차원 변수의 하위 집합으로 옴)을 사용하여 가능한 전체 값의 하위 집합을 만듭니다.

제한 사항:

데이터 쌍의 한쪽 또는 양쪽에 텍스트 값, NULL 값, 누락된 값이 있으면 전체 데이터 쌍이 무시됩니다.

예 및 결과:

예제 스크립트를 앱에 추가하고 실행합니다. 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.

함수 예

예	결과
mutualinfo(Age, Salary, 1)	차원 Employee name와 측정값 mutualinfo(Age, Salary, 1)가 포함된 테이블의 경우, 결과가 0.99820986입니다. 결과는 합계 셀에만 표시됩니다.
mutualinfo (TOTAL Age, Salary, 1, null (), 81)	Gender 차원이 있는 필터 창을 만들고 선택 작업을 수행하면 Female을 선택한 경우 0.99805677이, Male을 선택한 경우 0.99847373이 결과로 표시됩니다. 이는 Gender의 다른 값에 속하지 않는 모든 결과가 선택에서 제외되기 때문입니다.
mutualinfo (TOTAL Age, Gender, 1, ValueLoop (25,35))	0.68196996. Gender에서 값을 선택하면 이 값이 0으로 변경됩니다.
mutualinfo({1} TOTAL Age, Salary, 1, null ())	0.99820986. 이것은 선택과 관계가 없습니다. 집합 표현식 {1}이 모든 선택 및 차원을 무시합니다.

예에서 사용된 데이터:

```
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()는 **group by** 절로 정의된 여러 레코드에서 표현식의 왜곡도를 반환합니다.

구문:

```
Skew([ distinct] expr)
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
expr	측정할 데이터가 포함된 표현식 또는 필드입니다.
DISTINCT	표현식 앞에 **distinct**라는 단어가 있을 경우 모든 중복 항목이 무시됩니다.

예 및 결과:

예제 스크립트를 앱에 추가하고 실행합니다. 그런 다음, `Type` 및 *MySkew* 를 차원으로 사용하는 일반표를 생성합니다.

결과 데이터

예	결과

`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;`
	Skew() 계산 결과는 다음과 같습니다.
	• `Type` 은 `MySkew`입니다.
	• `Comparison` 은 0.86414768입니다.
	• `Observation` 은 0.32625351입니다.

Skew - 차트 함수

Skew()는 차트 차원에서 반복되는 표현식 또는 필드의 집계된 왜곡도를 반환합니다.

구문:

```
Skew([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr)
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
expr	측정할 데이터가 포함된 표현식 또는 필드입니다.
SetExpression	기본적으로 집계 함수는 선택에 의해 정의된 사용 가능한 레코드의 집합을 집계합니다. 집합 분석 표현식으로 대체 레코드 집합을 정의할 수 있습니다.
DISTINCT	함수 인수 앞에 **DISTINCT**라는 단어가 있을 경우 해당 함수 인수의 평가 결과로 생성된 중복이 무시됩니다.
TOTAL	**TOTAL**이 함수 인수 앞에 오는 경우, 현재 선택을 고려하되 현재 차원 값에 관련되지 않은 가능한 모든 값에 대한 계산이 실행됩니다. 즉, 차트 차원은 무시됩니다. **TOTAL [<fld {.fld}>]**(여기서 **TOTAL** 한정자 뒤에는 하나 이상의 필드 이름 목록이 차트 차원 변수의 하위 집합으로 옴)을 사용하여 가능한 전체 값의 하위 집합을 만듭니다.

제한 사항:

집계 함수의 매개 변수는 이러한 내부 집계에 **TOTAL** 한정자가 포함되어 있지 않는 한 다른 집계 함수를 포함하지 않아야 합니다. 고급 중첩 집계가 필요한 경우는 고급 함수 **Aggr**을 지정된 차원과 함께 사용하십시오.

예 및 결과:

예제 스크립트를 앱에 추가하고 실행합니다. 그런 다음, `Type` 을 차원으로, `Skew(Value)`를 측정값으로 사용하는 일반표를 생성합니다.

`Totals`는 테이블 속성에서 활성화해야 합니다.

예	결과
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 '|');	Skew(Value) 계산 결과는 다음과 같습니다. • Total은 0.23522195입니다. • Comparison 은 0.86414768입니다. • Observation 은 0.32625351입니다.

관련 항목:

Avg - 차트 함수 (page 389)

Stdev

Stdev()는 **group by** 절로 정의된 여러 레코드에서 표현식으로 지정한 값의 표준 편차를 반환합니다.

구문:

```
Stdev([distinct] expr)
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
expr	측정할 데이터가 포함된 표현식 또는 필드입니다.
distinct	표현식 앞에 **distinct**라는 단어가 있을 경우 모든 중복 항목이 무시됩니다.

예 및 결과:

예제 스크립트를 앱에 추가하고 실행합니다. 그런 다음, Type 및 MyStdev를 차원으로 사용하는 일반표를 생성합니다.

결과 데이터

예	결과
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;	Stdev() 계산 결과는 다음과 같습니다. • Type 은 MyStdev입니다. • Comparison 은 14.61245입니다. • Observation 은 12.507997입니다.

Stdev - 차트 함수

Stdev()는 차트 차원에서 반복되는 표현식 또는 필드에서 집계된 데이터 범위의 표준 편차를 찾습니다.

구문:

```
Stdev([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr)
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
expr	측정할 데이터가 포함된 표현식 또는 필드입니다.
SetExpression	기본적으로 집계 함수는 선택에 의해 정의된 사용 가능한 레코드의 집합을 집계합니다. 집합 분석 표현식으로 대체 레코드 집합을 정의할 수 있습니다.
DISTINCT	함수 인수 앞에 **DISTINCT**라는 단어가 있을 경우 해당 함수 인수의 평가 결과로 생성된 중복이 무시됩니다.
TOTAL	**TOTAL**이 함수 인수 앞에 오는 경우, 현재 선택을 고려하되 현재 차원 값에 관련되지 않은 가능한 모든 값에 대한 계산이 실행됩니다. 즉, 차트 차원은 무시됩니다. **TOTAL [<fld {.fld}>]**(여기서 **TOTAL** 한정자 뒤에는 하나 이상의 필드 이름 목록이 차트 차원 변수의 하위 집합으로 옴)을 사용하여 가능한 전체 값의 하위 집합을 만듭니다.

제한 사항:

집계 함수의 매개 변수는 이러한 내부 집계에 **TOTAL** 한정자가 포함되어 있지 않는 한 다른 집계 함수를 포함하지 않아야 합니다. 고급 중첩 집계가 필요한 경우는 고급 함수 **Aggr**을 지정된 차원과 함께 사용하십시오.

예 및 결과:

예제 스크립트를 앱에 추가하고 실행합니다. 그런 다음, `Type` 을 차원으로, `Stdev(Value)`를 측정값으로 사용하는 일반표를 생성합니다.

`Totals`는 테이블 속성에서 활성화해야 합니다.

예	결과
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 '|');	Stdev(Value) 계산 결과는 다음과 같습니다. • Total은 15.47529입니다. • Comparison 은 14.61245입니다. • Observation 은 12.507997입니다.

관련 항목:

- *Avg - 차트 함수 (page 389)*
- *STEYX - 차트 함수 (page 442)*

Sterr

Sterr()은 **group by** 절로 정의된 여러 레코드에서 반복되는 표현식으로 표현된 일련의 값에 대해 집계된 표준 오차(stdev/sqrt(n))를 반환합니다.

구문:

```
Sterr ([distinct] expr)
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
expr	측정할 데이터가 포함된 표현식 또는 필드입니다.
distinct	표현식 앞에 **distinct**라는 단어가 있을 경우 모든 중복 항목이 무시됩니다.

제한 사항:

텍스트 값, NULL 값, 누락된 값은 무시됩니다.

예 및 결과:

예제 스크립트를 앱에 추가하고 실행합니다. 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.

결과 데이터

예	결과
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;	차원 Type 및 MySterr이 있는 테이블에서 데이터 로드 스크립트의 Sterr() 계산 결과는 다음과 같습니다. Type MySterr Comparison 3.2674431 Observation 2.7968733

Sterr - 차트 함수

Sterr()는 차트 차원에서 반복되는 표현식에서 집계된 일련의 값에 대한 평균값의 표준 오차 값 (stdev/sqrt (n))을 찾습니다.

구문:

```
Sterr([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] expr)
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
expr	측정할 데이터가 포함된 표현식 또는 필드입니다.
SetExpression	기본적으로 집계 함수는 선택에 의해 정의된 사용 가능한 레코드의 집합을 집계합니다. 집합 분석 표현식으로 대체 레코드 집합을 정의할 수 있습니다.
DISTINCT	함수 인수 앞에 **DISTINCT**라는 단어가 있을 경우 해당 함수 인수의 평가 결과로 생성된 중복이 무시됩니다.
TOTAL	**TOTAL**이 함수 인수 앞에 오는 경우, 현재 선택을 고려하되 현재 차원 값에 관련되지 않은 가능한 모든 값에 대한 계산이 실행됩니다. 즉, 차트 차원은 무시됩니다. **TOTAL [<fld {.fld}>]**(여기서 **TOTAL** 한정자 뒤에는 하나 이상의 필드 이름 목록이 차트 차원 변수의 하위 집합으로 옴)을 사용하여 가능한 전체 값의 하위 집합을 만듭니다.

제한 사항:

집계 함수의 매개 변수는 이러한 내부 집계에 **TOTAL** 한정자가 포함되어 있지 않는 한 다른 집계 함수를 포함하지 않아야 합니다. 고급 중첩 집계가 필요한 경우는 고급 함수 **Aggr**을 지정된 차원과 함께 사용하십시오.

텍스트 값, NULL 값, 누락된 값은 무시됩니다.

예 및 결과:

예제 스크립트를 앱에 추가하고 실행합니다. 그런 다음, `Type` 을 차원으로, `Sterr(Value)`를 측정값으로 사용하는 일반표를 생성합니다.

`Totals`는 테이블 속성에서 활성화해야 합니다.

예	결과
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 '|');	Sterr(Value) 계산 결과는 다음과 같습니다. • Total은 2.4468583입니다. • Comparison 은 3.2674431입니다. • Observation 은 2.7968733입니다.

관련 항목:

Avg - 차트 함수 (page 389)
STEYX - 차트 함수 (page 442)

STEYX

STEYX()는 **group by** 절로 정의된 여러 레코드에서 반복되는 x-expression 및 y-expression의 숫자 쌍으로 표현된 일련의 좌표에 대한 회귀 내의 각 x 값에 대해 예측된 y 값의 집계된 표준 오차를 반환합니다.

구문:

```
STEYX (y_value, x_value)
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
y_value	y 값 범위를 포함하는 표현식 또는 필드가 측정됩니다.
x_value	x 값 범위를 포함하는 표현식 또는 필드가 측정됩니다.

제한 사항:

데이터 쌍의 한쪽 또는 양쪽에 텍스트 값, NULL 값, 누락된 값이 있으면 전체 데이터 쌍이 무시됩니다.

예 및 결과:

예제 스크립트를 앱에 추가하고 실행합니다. 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.

결과 데이터

예	결과
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;	차원 MySTEYX가 있는 테이블에서 데이터 로드 스크립트의 STEYX() 계산 결과는 2.0714764입니다.

STEYX - 차트 함수

STEYX()는 **y_value** 및 **x_value**를 통해 지정된 표현식의 숫자 쌍으로 표현된 일련의 좌표에 의해 지정된 선형 회귀의 각 x 값에 대해 y 값을 예측하는 경우의 집계된 표준 오차를 반환합니다.

구문:

```
STEYX([{SetExpression}] [DISTINCT] [TOTAL [<fld{, fld}>]] y_value, x_value)
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
y_value	측정할 알려진 y 값의 범위가 포함된 표현식 또는 필드.
x_value	측정할 알려진 x 값의 범위가 포함된 표현식 또는 필드.
SetExpression	기본적으로 집계 함수는 선택에 의해 정의된 사용 가능한 레코드의 집합을 집계합니다. 집합 분석 표현식으로 대체 레코드 집합을 정의할 수 있습니다.
DISTINCT	함수 인수 앞에 **DISTINCT**라는 단어가 있을 경우 해당 함수 인수의 평가 결과로 생성된 중복이 무시됩니다.
TOTAL	**TOTAL**이 함수 인수 앞에 오는 경우, 현재 선택을 고려하되 현재 차원 값에 관련되지 않은 가능한 모든 값에 대한 계산이 실행됩니다. 즉, 차트 차원은 무시됩니다. **TOTAL [<fld {.fld}>]**(여기서 **TOTAL** 한정자 뒤에는 하나 이상의 필드 이름 목록이 차트 차원 변수의 하위 집합으로 옴)을 사용하여 가능한 전체 값의 하위 집합을 만듭니다.

제한 사항:

집계 함수의 매개 변수는 이러한 내부 집계에 **TOTAL** 한정자가 포함되어 있지 않는 한 다른 집계 함수를 포함하지 않아야 합니다. 고급 중첩 집계가 필요한 경우는 고급 함수 **Aggr**을 지정된 차원과 함께 사용하십시오.

데이터 쌍의 한쪽 또는 양쪽에 텍스트 값, NULL 값, 누락된 값이 있으면 전체 데이터 쌍이 무시됩니다.

예 및 결과:

예제 스크립트를 앱에 추가하고 실행합니다. 그런 다음, `KnownY` 및 `KnownX` 를 차원으로, `Steyx(KnownY,KnownX)`를 측정값으로 사용하는 일반표를 생성합니다.

`Totals`는 테이블 속성에서 활성화해야 합니다.

예	결과
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 '|');	STEYX(KnownY,KnownX) 계산 결과는 2.071입니다(숫자 서식이 십진수 3자리로 설정된 경우).

관련 항목:

- *Avg - 차트 함수 (page 389)*
- *Sterr - 차트 함수 (page 438)*

linest 함수를 사용하는 방법의 예

linest 함수는 선형 회귀 분석과 관련된 값을 찾는 데 사용됩니다. 이 섹션에서는 샘플 데이터를 사용하여 시각화를 작성하고 linest에서 사용 가능한 Qlik Sense 함수의 값을 찾는 방법을 설명합니다. linest 함수는 데이터 로드 스크립트와 차트 표현식에서 사용할 수 있습니다.

구문과 인수에 대한 설명은 각 linest 차트 함수 및 스크립트 함수 항목을 참조하십시오.

예에 사용된 데이터 및 스크립트 표현식

아래의 linest() 예에 대한 데이터 로드 편집기에서 다음 인라인 데이터 및 스크립트 표현식을 로드합니다.

```
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;
```

예 1: linest를 사용하는 스크립트 표현식

예: 스크립트 표현식

데이터 로드 스크립트 계산에서 시각화 만들기

다음 필드를 열로 사용하여 Qlik Sense 시트에 테이블 시각화를 만듭니다.

- Linest_B
- Linest_DF
- Linest_F
- Linest_M

- Linest_R2
- Linest_SEB
- Linest_SEM
- Linest_SEY
- Linest_SSREG
- Linest_SSRESID

결과

데이터 로드 스크립트에서 생성된 linest 계산의 결과가 포함된 테이블은 다음과 같이 표시됩니다.

결과 테이블

Linest_B	Linest_DF	Linest_F	Linest_M	Linest_R2	Linest_SEB
-35.047	18	20.788	8.605	0.536	22.607

결과 테이블

Linest_SEM	Linest_SEY	Linest_SSREG	Linest_SSRESID
1.887	48.666	49235.014	42631.186

예 2: linest를 사용하는 차트 표현식

예: 차트 표현식

다음 필드를 차원으로 사용하여 Qlik Sense 시트에 테이블 시각화를 만듭니다.

```
ValueList('Linest_b', 'Linest_df','Linest_f', 'Linest_m','Linest_r2','Linest_SEB','Linest_
SEM','Linest_SEY','Linest_SSREG','Linest_SSRESID')
```

이 표현식은 가상 차원 함수를 사용하여 linest 함수 이름이 포함된 차원에 대한 레이블을 만듭니다. 공간을 절약하기 위해 레이블을 **Linest functions**로 변경할 수 있습니다.

테이블에 다음 표현식을 측정값으로 추가합니다.

```
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) )
```

이 표현식은 가상 차원에 있는 해당 이름에 대응하는 각 linest 함수의 결과 값이 표시됩니다. `Linest_b(Y,X)`의 결과가 **linest_b** 옆에 표시됩니다.

결과

결과 테이블

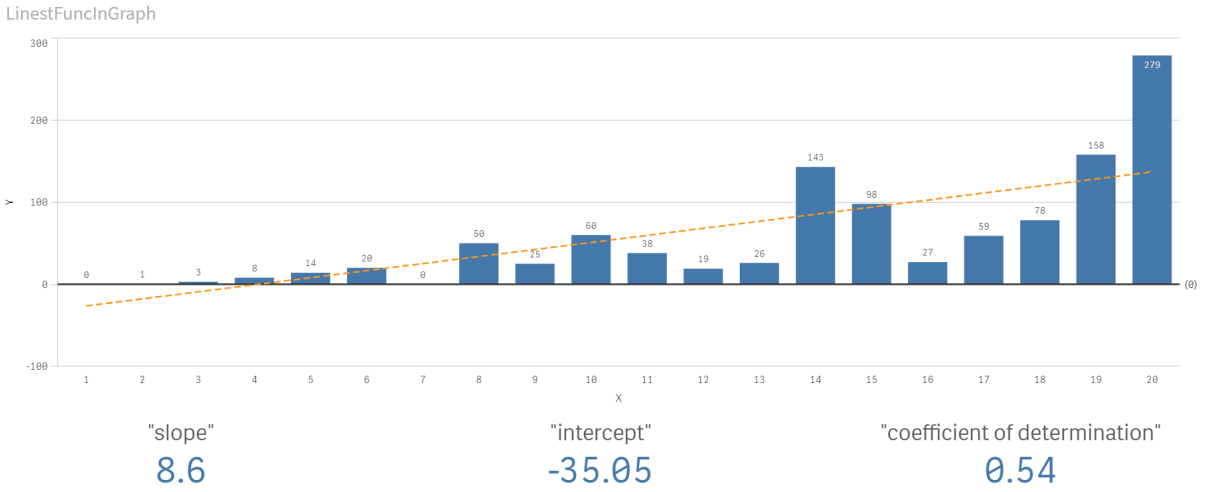
Linest functions	**Linest function results**
Linest_b	-35.047
Linest_df	18
Linest_f	20.788
Linest_m	8.605
Linest_r2	0.536
Linest_SEB	22.607
Linest_SEM	1.887
Linest_SEY	48.666
Linest_SSREG	49235.014
Linest_SSRESID	42631.186

예 3: linest를 사용하는 차트 표현식

예: 차트 표현식

1. **X**를 차원으로, **Y**를 측정값으로 사용하여 Qlik Sense 시트에 막대 차트 시각화를 만듭니다.
2. Y 측정값에 선형 추세선을 추가합니다.
3. 시트에 KPI 시각화를 추가합니다.
    1. KPI에 대한 레이블로 *slope*을 추가합니다.
    2. KPI에 대한 표현식으로 `sum(Linest_M)`을 추가합니다.
4. 시트에 두 번째 KPI 시각화를 추가합니다.
    1. KPI에 대한 레이블로 *intercept*를 추가합니다.
    2. KPI에 대한 표현식으로 `Sum(Linest_B)`을 추가합니다.
5. 시트에 세 번째 KPI 시각화를 추가합니다.
    1. KPI에 대한 레이블로 *coefficient of determination*을 추가합니다.
    2. KPI에 대한 표현식으로 `Sum(Linest_R2)`을 추가합니다.

결과



설명

막대 차트는 X 및 Y 데이터를 표시합니다. 관련 linest() 함수는 추세선의 기반이 되는 선형 회귀 방정식, 즉 y = m * x + b에 대한 값을 제공합니다. 방정식은 "최소 제곱" 방법을 사용하여 데이터에 가장 적합한 선을 설명하는 배열을 반환하여 직선(추세 선)을 계산합니다.

KPI는 선형 회귀 방정식의 변수가 되는 기울기에 대한 linest() 함수 **sum(Linest_M)** 및 Y 절편의 경우 **sum(Linest_B)**의 결과 및 결정 계수에 대한 해당 집계 R2 값을 표시합니다.

통계 테스트 함수

통계 테스트 함수는 데이터 로드 스크립트와 차트 표현식에서 모두 사용할 수 있지만 구문이 다릅니다.

Chi-2 검정 함수

일반적으로 정성적 변수의 연구에 사용됩니다. 예상된 빈도를 갖는 일원 빈도 테이블에서 관찰된 빈도를 비교하거나 분할표에서 두 변수 사이의 관계를 연구할 수 있습니다.

t 검정 함수

t 검정 함수는 두 모평균의 통계 시험에 사용됩니다. 두 표본 t 검정은 두 표본이 서로 다른지 시험하며 두 정규 분포가 알려지지 않은 분포일 경우 및 실험에 작은 크기의 표본을 사용할 경우에 주로 사용됩니다.

z 검정 함수

두 모평균의 통계 시험입니다. 두 표본 z 검정은 두 표본이 서로 다른지 시험하며 두 정규 분포가 알려진 분포일 경우 및 실험에 큰 크기의 표본을 사용할 경우에 주로 사용됩니다.

Chi2-test 함수

일반적으로 정성적 변수의 연구에 사용됩니다. 예상된 빈도를 갖는 일원 빈도 테이블에서 관찰된 빈도를 비교하거나 분할표에서 두 변수 사이의 관계를 연구할 수 있습니다.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.

이 함수를 데이터 로드 스크립트에서 사용하면 group by 절로 정의된 여러 레코드에서 값이 반복됩니다.

차트 표현식에서 이 함수를 사용하면 차트 차원에 대해 값이 반복됩니다.

Chi2Test_chi2
Chi2Test_chi2()는 하나 또는 두 수열에 대해 집계된 chi^2-검정 값을 반환합니다.

```
Chi2Test_chi2()는 하나 또는 두 수열에 대해 집계된 chi2-검정 값을 반환합니다.(col, row, actual_value[, expected_value])
```

Chi2Test_df
Chi2Test_df()는 하나 또는 두 수열에 대해 집계된 chi^2 검정 df 값(자유도)을 반환합니다.

```
Chi2Test_df()는 하나 또는 두 수열에 대해 집계된 chi2 검정 df 값(자유도)을 반환합니다.(col, row, actual_value[, expected_value])
```

Chi2Test_p
Chi2Test_p()는 하나 또는 두 수열에 대해 집계된 chi^2 검정 p 값(유의성)을 반환합니다.

```
Chi2Test_p - 차트 함수(col, row, actual_value[, expected_value])
```

관련 항목:

- *T-test 함수 (page 451)*
- *Z-test 함수 (page 484)*

Chi2Test_chi2

Chi2Test_chi2()는 하나 또는 두 수열에 대해 집계된 chi^2-검정 값을 반환합니다.

이 함수를 데이터 로드 스크립트에서 사용하면 group by 절로 정의된 여러 레코드에서 값이 반복됩니다.

차트 표현식에서 이 함수를 사용하면 차트 차원에 대해 값이 반복됩니다.

모든 Qlik Sense chi^2 검정 함수는 동일한 인수를 갖습니다.

구문:

```
Chi2Test_chi2(col, row, actual_value[, expected_value])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
col, row	값의 행렬에서 지정한 열 및 행이 테스트됩니다.
actual_value	지정된 **col** 및 **row**에서 관찰된 데이터의 값입니다.
expected_value	지정된 **col** 및 **row**에서 분포에 대한 예상 값입니다.

제한 사항:

표현식 값에 텍스트 값, NULL 값, 누락된 값이 있으면 함수에서 NULL을 반환합니다.

```
Chi2Test_chi2( Grp, Grade, Count )
Chi2Test_chi2( Gender, Description, Observed, Expected )
```

관련 항목:

- *차트에 chi2-test 함수를 사용하는 방법의 예 (page 497)*
- *데이터 로드 스크립트에서 chi2-test 함수를 사용하는 방법의 예 (page 501)*

Chi2Test_df

Chi2Test_df()는 하나 또는 두 수열에 대해 집계된 chi^2 검정 df 값(자유도)을 반환합니다.

이 함수를 데이터 로드 스크립트에서 사용하면 group by 절로 정의된 여러 레코드에서 값이 반복됩니다.

차트 표현식에서 이 함수를 사용하면 차트 차원에 대해 값이 반복됩니다.

> *모든 Qlik Sense chi^2 검정 함수는 동일한 인수를 갖습니다.*

구문:

```
Chi2Test_df(col, row, actual_value[, expected_value])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
col, row	값의 행렬에서 지정한 열 및 행이 테스트됩니다.
actual_value	지정된 **col** 및 **row**에서 관찰된 데이터의 값입니다.
expected_value	지정된 **col** 및 **row**에서 분포에 대한 예상 값입니다.

제한 사항:

표현식 값에 텍스트 값, NULL 값, 누락된 값이 있으면 함수에서 NULL을 반환합니다.

```
Chi2Test_df( Grp, Grade, Count )
Chi2Test_df( Gender, Description, Observed, Expected )
```

관련 항목:

- *차트에 chi2-test 함수를 사용하는 방법의 예 (page 497)*
- *데이터 로드 스크립트에서 chi2-test 함수를 사용하는 방법의 예 (page 501)*

Chi2Test_p - 차트 함수

Chi2Test_p()는 하나 또는 두 수열에 대해 집계된 chi^2 검정 p 값(유의성)을 반환합니다. 테스트는 지정된 **col** 및 **row** 행렬 내의 변이에 대한 **actual_value** 검정 값에 대해 수행하거나 **actual_value**의 값과 지정된 경우 **expected_value**의 해당 값을 비교하여 수행할 수 있습니다.

이 함수를 데이터 로드 스크립트에서 사용하면 group by 절로 정의된 여러 레코드에서 값이 반복됩니다.

차트 표현식에서 이 함수를 사용하면 차트 차원에 대해 값이 반복됩니다.

> *모든 Qlik Sense chi^2 검정 함수는 동일한 인수를 갖습니다.*

구문:

```
Chi2Test_p(col, row, actual_value[, expected_value])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
col, row	값의 행렬에서 지정한 열 및 행이 테스트됩니다.
actual_value	지정된 **col** 및 **row**에서 관찰된 데이터의 값입니다.
expected_value	지정된 **col** 및 **row**에서 분포에 대한 예상 값입니다.

제한 사항:

표현식 값에 텍스트 값, NULL 값, 누락된 값이 있으면 함수에서 NULL을 반환합니다.

```
Chi2Test_p( Grp, Grade, Count )
Chi2Test_p( Gender, Description, Observed, Expected )
```

관련 항목:

차트에 chi2-test 함수를 사용하는 방법의 예 (page 497)
데이터 로드 스크립트에서 chi2-test 함수를 사용하는 방법의 예 (page 501)

T-test 함수

t 검정 함수는 두 모평균의 통계 시험에 사용됩니다. 두 표본 t 검정은 두 표본이 서로 다른지 시험하며 두 정규 분포가 알려지지 않은 분포일 경우 및 실험에 작은 크기의 표본을 사용할 경우에 주로 사용됩니다.

다음 섹션에는 T 검정 통계 검정 함수가 각 함수 유형에 적용되는 샘플 학생 테스트에 따라 그룹화되어 있습니다.

일반적인 t-test 보고서 만들기 (page 503)

두 독립 표본 T 검정

다음 함수가 두 독립 표본 학생 T 검정에 적용됩니다.

ttest_conf

TTest_conf는 두 독립 수열에 대해 집계된 t 신뢰 구간 값을 반환합니다.

`TTest_conf는 두 독립 수열에 대해 집계된 t 신뢰 구간 값을 반환합니다. (` `grp, value [, sig[, eq_var]]` **`)`**

ttest_df

TTest_df()는 두 독립 수열에 대해 집계된 학생 t 검정 값(자유도)을 반환합니다.

TTest_df()는 두 독립 수열에 대해 집계된 학생 t 검정 값(자유도)을 반환합니다. (grp, value [, eq_var)

ttest_dif

TTest_dif()는 두 독립 수열에 대해 집계된 학생 t 검정 평균값 차를 반환하는 수치 함수입니다.

TTest_dif()는 두 독립 수열에 대해 집계된 학생 t 검정 평균값 차를 반환하는 수치 함수입니다. (grp, value)

ttest_lower

TTest_lower()는 두 독립 수열에 대한 신뢰 구간의 하단에 대해 집계된 값을 반환합니다.

TTest_lower()는 두 독립 수열에 대한 신뢰 구간의 하단에 대해 집계된 값을 반환합니다. (grp, value [, sig[, eq_var]])

ttest_sig

TTest_sig()는 두 독립 수열에 대해 집계된 학생 t 검정 양측 유의성 수준을 반환합니다.

TTest_sig()는 두 독립 수열에 대해 집계된 학생 t 검정 양측 유의성 수준을 반환합니다. (grp, value [, eq_var])

ttest_sterr

TTest_sterr()은 두 독립 수열에 대해 집계된 학생 t 검정 평균값 차의 표준 오차를 반환합니다.

TTest_sterr()은 두 독립 수열에 대해 집계된 학생 t 검정 평균값 차의 표준 오차를 반환합니다. (grp, value [, eq_var])

ttest_t

TTest_t()는 두 독립 수열에 대해 집계된 t 값을 반환합니다.

TTest_t()는 두 독립 수열에 대해 집계된 t 값을 반환합니다. (grp, value [, eq_var])

ttest_upper

TTest_upper()는 두 독립 수열에 대한 신뢰 구간의 상단에 대해 집계된 값을 반환합니다.

TTest_upper()는 두 독립 수열에 대한 신뢰 구간의 상단에 대해 집계된 값을 반환합니다. (grp, value [, sig [, eq_var]])

두 독립 가중 표본 T 검정

다음 함수가 두 독립 표본 학생 T 검정에 적용되며, 여기서 입력 데이터 수열은 가중치가 적용된 2열 형식으로 지정됩니다.

ttestw_conf

TTestw_conf()는 두 독립 수열에 대해 집계된 t 값을 반환합니다.

TTestw_conf()는 두 독립 수열에 대해 집계된 t 값을 반환합니다. (weight, grp, value [, sig[, eq_var]])

ttestw_df

TTestw_df()는 두 독립 수열에 대해 집계된 학생 t 검정 df 값(자유도)을 반환합니다.

TTestw_df()는 두 독립 수열에 대해 집계된 학생 t 검정 df 값(자유도)을 반환합니다.
(weight, grp, value [, eq_var]**)**

ttestw_dif
TTestw_dif()는 두 독립 수열에 대해 집계된 학생 t 검정 평균값 차를 반환합니다.

TTestw_dif()는 두 독립 수열에 대해 집계된 학생 t 검정 평균값 차를 반환합니다. (
weight, grp, value**)**

ttestw_lower
TTestw_lower()는 두 독립 수열에 대한 신뢰 구간의 하단에 대해 집계된 값을 반환합니다.

TTestw_lower()는 두 독립 수열에 대한 신뢰 구간의 하단에 대해 집계된 값을 반환합니다.
(weight, grp, value [, sig[, eq_var]]**)**

ttestw_sig
TTestw_sig()는 두 독립 수열에 대해 집계된 학생 t 검정 양측 유의성 수준을 반환합니다.

TTestw_sig()는 두 독립 수열에 대해 집계된 학생 t 검정 양측 유의성 수준을 반환합니다. (
weight, grp, value [, eq_var]**)**

ttestw_sterr
TTestw_sterr()은 두 독립 수열에 대해 집계된 학생 t 검정 평균값 차의 표준 오차를 반환합니다.

TTestw_sterr()은 두 독립 수열에 대해 집계된 학생 t 검정 평균값 차의 표준 오차를 반환합니다. (weight, grp, value [, eq_var]**)**

ttestw_t
TTestw_t()는 두 독립 수열에 대해 집계된 t 값을 반환합니다.

TTestw_t()는 두 독립 수열에 대해 집계된 t 값을 반환합니다. (weight, grp, value [, eq_var]**)**

ttestw_upper
TTestw_upper()는 두 독립 수열에 대한 신뢰 구간의 상단에 대해 집계된 값을 반환합니다.

TTestw_upper()는 두 독립 수열에 대한 신뢰 구간의 상단에 대해 집계된 값을 반환합니다.
(weight, grp, value [, sig [, eq_var]]**)**

단일 표본 T 검정

다음 함수가 단일 표본 학생 T 검정에 적용됩니다.

ttest1_conf
TTest1_conf()는 수열에 대해 집계된 신뢰 구간 값을 반환합니다.

TTest1_conf()는 수열에 대해 집계된 신뢰 구간 값을 반환합니다. (value [, sig]**)**

ttest1_df
TTest1_df()는 수열에 대해 집계된 학생 t 검정 df 값(자유도)을 반환합니다.

TTest1_df()는 수열에 대해 집계된 학생 t 검정 df 값(자유도)을 반환합니다. (value**)**

ttest1_dif
TTest1_dif()는 수열에 대해 집계된 학생 t 검정 평균값 차를 반환합니다.

TTest1_dif()는 수열에 대해 집계된 학생 t 검정 평균값 차를 반환합니다. (value)

ttest1_lower
TTest1_lower()는 수열에 대한 신뢰 구간의 하단에 대해 집계된 값을 반환합니다.

TTest1_lower()는 수열에 대한 신뢰 구간의 하단에 대해 집계된 값을 반환합니다. (value [, sig])

ttest1_sig
TTest1_sig()는 수열에 대해 집계된 학생 t 검정 양측 유의성 수준을 반환합니다.

TTest1_sig()는 수열에 대해 집계된 학생 t 검정 양측 유의성 수준을 반환합니다. (value)

ttest1_sterr
TTest1_sterr()은 수열에 대해 집계된 학생 t 검정 평균값 차의 표준 오차를 반환합니다.

TTest1_sterr()은 수열에 대해 집계된 학생 t 검정 평균값 차의 표준 오차를 반환합니다. (value)

ttest1_t
TTest1_t() 는 수열에 대해 집계된 t 값을 반환합니다.

TTest1_t() 는 수열에 대해 집계된 t 값을 반환합니다. (value)

ttest1_upper
TTest1_upper()는 수열에 대한 신뢰 구간의 상단에 대해 집계된 값을 반환합니다.

TTest1_upper()는 수열에 대한 신뢰 구간의 상단에 대해 집계된 값을 반환합니다. (value [, sig])

단일 가중 표본 T 검정

다음 함수가 단일 표본 학생 T 검정에 적용되며, 여기서 입력 데이터 수열은 가중치가 적용된 2열 형식으로 지정됩니다.

ttest1w_conf
TTest1w_conf()는 수열에 대해 집계된 신뢰 구간 값을 반환하는 **숫자** 함수입니다.

TTest1w_conf()는 수열에 대해 집계된 신뢰 구간 값을 반환하는 숫자 함수입니다. (weight, value [, sig])

ttest1w_df
TTest1w_df()는 수열에 대해 집계된 학생 t 검정 df 값(자유도)을 반환합니다.

TTest1w_df()는 수열에 대해 집계된 학생 t 검정 df 값(자유도)을 반환합니다. (weight, value)

ttest1w_dif
TTest1w_dif()는 수열에 대해 집계된 학생 t 검정 평균값 차를 반환합니다.

TTest1w_dif()는 수열에 대해 집계된 학생 t 검정 평균값 차를 반환합니다. (weight, value)

ttest1w_lower

TTest1w_lower()는 수열에 대한 신뢰 구간의 하단에 대해 집계된 값을 반환합니다.

TTest1w_lower()는 수열에 대한 신뢰 구간의 하단에 대해 집계된 값을 반환합니다. (weight, value [, sig])

ttest1w_sig

TTest1w_sig()는 수열에 대해 집계된 학생 t 검정 양측 유의성 수준을 반환합니다.

TTest1w_sig()는 수열에 대해 집계된 학생 t 검정 양측 유의성 수준을 반환합니다. (weight, value)

ttest1w_sterr

TTest1w_sterr()은 수열에 대해 집계된 학생 t 검정 평균값 차의 표준 오차를 반환합니다.

TTest1w_sterr()은 수열에 대해 집계된 학생 t 검정 평균값 차의 표준 오차를 반환합니다. (weight, value)

ttest1w_t

TTest1w_t() 는 수열에 대해 집계된 t 값을 반환합니다.

TTest1w_t() 는 수열에 대해 집계된 t 값을 반환합니다. (weight, value)

ttest1w_upper

TTest1w_upper()는 수열에 대한 신뢰 구간의 상단에 대해 집계된 값을 반환합니다.

TTest1w_upper()는 수열에 대한 신뢰 구간의 상단에 대해 집계된 값을 반환합니다. (weight, value [, sig])

TTest_conf

TTest_conf는 두 독립 수열에 대해 집계된 t 신뢰 구간 값을 반환합니다.

이 함수는 독립 표본 학생 t 검정에 적용됩니다.

이 함수를 데이터 로드 스크립트에서 사용하면 group by 절로 정의된 여러 레코드에서 값이 반복됩니다.

차트 표현식에서 이 함수를 사용하면 차트 차원에 대해 값이 반복됩니다.

구문:

TTest_conf (grp, value [, sig [, eq_var]])

반환 데이터 유형: 숫자

인수:

인수

인수	설명
value	평가할 표본 값입니다. 표본 값은 **group**의 두 값으로 정확하게 지정하여 논리적으로 그룹화되어야 합니다. 로드 스크립트에서 표본 값에 대해 필드 이름을 지정하지 않으면 필드 이름으로 **Value**가 자동 지정됩니다.
grp	두 표본 그룹의 이름이 각각 포함된 필드입니다. 로드 스크립트에서 그룹에 대해 필드 이름을 지정하지 않으면 필드 이름으로 **Type**이 자동 지정됩니다.
sig	양측 유의성 수준은 **sig**에 지정할 수 있습니다. 생략할 경우 **sig**는 0.025로 설정되며, 신뢰 구간은 95%가 됩니다.
eq_var	**eq_var**이 False(0)로 지정된 경우, 두 표본의 개별 분산으로 가정됩니다. **eq_var**이 True(1)로 지정된 경우, 표본 간의 등분산으로 가정됩니다.

제한 사항:

표현식 값에 텍스트 값, NULL 값, 누락된 값이 있으면 함수에서 NULL을 반환합니다.

```
TTest_conf( Group, Value )
TTest_conf( Group, Value, Sig, false )
```

관련 항목:

일반적인 t-test 보고서 만들기 (page 503)

TTest_df

TTest_df()는 두 독립 수열에 대해 집계된 학생 t 검정 값(자유도)을 반환합니다.

이 함수는 독립 표본 학생 t 검정에 적용됩니다.

이 함수를 데이터 로드 스크립트에서 사용하면 group by 절로 정의된 여러 레코드에서 값이 반복됩니다.

차트 표현식에서 이 함수를 사용하면 차트 차원에 대해 값이 반복됩니다.

구문:

```
TTest_df (grp, value [, eq_var])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
value	평가할 표본 값입니다. 표본 값은 **group**의 두 값으로 정확하게 지정하여 논리적으로 그룹화되어야 합니다. 로드 스크립트에서 표본 값에 대해 필드 이름을 지정하지 않으면 필드 이름으로 **Value**가 자동 지정됩니다.
grp	두 표본 그룹의 이름이 각각 포함된 필드입니다. 로드 스크립트에서 그룹에 대해 필드 이름을 지정하지 않으면 필드 이름으로 **Type**이 자동 지정됩니다.
eq_var	**eq_var**이 False(0)로 지정된 경우, 두 표본의 개별 분산으로 가정됩니다. **eq_var**이 True(1)로 지정된 경우, 표본 간의 등분산으로 가정됩니다.

제한 사항:

표현식 값에 텍스트 값, NULL 값, 누락된 값이 있으면 함수에서 NULL을 반환합니다.

```
TTest_df( Group, Value )
TTest_df( Group, Value, false )
```

관련 항목:

일반적인 t-test 보고서 만들기 (page 503)

TTest_dif

TTest_dif()는 두 독립 수열에 대해 집계된 학생 t 검정 평균값 차를 반환하는 수치 함수입니다.

이 함수는 독립 표본 학생 t 검정에 적용됩니다.

이 함수를 데이터 로드 스크립트에서 사용하면 group by 절로 정의된 여러 레코드에서 값이 반복됩니다.

차트 표현식에서 이 함수를 사용하면 차트 차원에 대해 값이 반복됩니다.

구문:

```
TTest_dif (grp, value [, eq_var] )
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
value	평가할 표본 값입니다. 표본 값은 **group**의 두 값으로 정확하게 지정하여 논리적으로 그룹화되어야 합니다. 로드 스크립트에서 표본 값에 대해 필드 이름을 지정하지 않으면 필드 이름으로 **Value**가 자동 지정됩니다.
grp	두 표본 그룹의 이름이 각각 포함된 필드입니다. 로드 스크립트에서 그룹에 대해 필드 이름을 지정하지 않으면 필드 이름으로 **Type**이 자동 지정됩니다.
eq_var	**eq_var**이 False(0)로 지정된 경우, 두 표본의 개별 분산으로 가정됩니다. **eq_var**이 True(1)로 지정된 경우, 표본 간의 등분산으로 가정됩니다.

제한 사항:

표현식 값에 텍스트 값, NULL 값, 누락된 값이 있으면 함수에서 NULL을 반환합니다.

```
TTest_dif( Group, Value )
TTest_dif( Group, Value, false )
```

관련 항목:

일반적인 t-test 보고서 만들기 (page 503)

TTest_lower

TTest_lower()는 두 독립 수열에 대한 신뢰 구간의 하단에 대해 집계된 값을 반환합니다.

이 함수는 독립 표본 학생 t 검정에 적용됩니다.

이 함수를 데이터 로드 스크립트에서 사용하면 group by 절로 정의된 여러 레코드에서 값이 반복됩니다.

차트 표현식에서 이 함수를 사용하면 차트 차원에 대해 값이 반복됩니다.

구문:

```
TTest_lower (grp, value [, sig [, eq_var]])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
value	평가할 표본 값입니다. 표본 값은 **group**의 두 값으로 정확하게 지정하여 논리적으로 그룹화되어야 합니다. 로드 스크립트에서 표본 값에 대해 필드 이름을 지정하지 않으면 필드 이름으로 **Value**가 자동 지정됩니다.
grp	두 표본 그룹의 이름이 각각 포함된 필드입니다. 로드 스크립트에서 그룹에 대해 필드 이름을 지정하지 않으면 필드 이름으로 **Type**이 자동 지정됩니다.
sig	양측 유의성 수준은 **sig**에 지정할 수 있습니다. 생략할 경우 **sig**는 0.025로 설정되며, 신뢰 구간은 95%가 됩니다.
eq_var	**eq_var**이 False(0)로 지정된 경우, 두 표본의 개별 분산으로 가정됩니다. **eq_var**이 True(1)로 지정된 경우, 표본 간의 등분산으로 가정됩니다.

제한 사항:

표현식 값에 텍스트 값, NULL 값, 누락된 값이 있으면 함수에서 NULL을 반환합니다.

```
TTest_lower( Group, Value )
TTest_lower( Group, Value, Sig, false )
```

관련 항목:

일반적인 t-test 보고서 만들기 (page 503)

TTest_sig

TTest_sig()는 두 독립 수열에 대해 집계된 학생 t 검정 양측 유의성 수준을 반환합니다.

이 함수는 독립 표본 학생 t 검정에 적용됩니다.

이 함수를 데이터 로드 스크립트에서 사용하면 group by 절로 정의된 여러 레코드에서 값이 반복됩니다.

차트 표현식에서 이 함수를 사용하면 차트 차원에 대해 값이 반복됩니다.

구문:

```
TTest_sig (grp, value [, eq_var])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
value	평가할 표본 값입니다. 표본 값은 **group**의 두 값으로 정확하게 지정하여 논리적으로 그룹화되어야 합니다. 로드 스크립트에서 표본 값에 대해 필드 이름을 지정하지 않으면 필드 이름으로 **Value**가 자동 지정됩니다.
grp	두 표본 그룹의 이름이 각각 포함된 필드입니다. 로드 스크립트에서 그룹에 대해 필드 이름을 지정하지 않으면 필드 이름으로 **Type**이 자동 지정됩니다.
eq_var	**eq_var**이 False(0)로 지정된 경우, 두 표본의 개별 분산으로 가정됩니다. **eq_var**이 True(1)로 지정된 경우, 표본 간의 등분산으로 가정됩니다.

제한 사항:

표현식 값에 텍스트 값, NULL 값, 누락된 값이 있으면 함수에서 NULL을 반환합니다.

```
TTest_sig( Group, Value )
TTest_sig( Group, Value, false )
```

관련 항목:

일반적인 t-test 보고서 만들기 (page 503)

TTest_sterr

TTest_sterr()은 두 독립 수열에 대해 집계된 학생 t 검정 평균값 차의 표준 오차를 반환합니다.

이 함수는 독립 표본 학생 t 검정에 적용됩니다.

이 함수를 데이터 로드 스크립트에서 사용하면 group by 절로 정의된 여러 레코드에서 값이 반복됩니다.

차트 표현식에서 이 함수를 사용하면 차트 차원에 대해 값이 반복됩니다.

구문:

```
TTest_sterr (grp, value [, eq_var])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
value	평가할 표본 값입니다. 표본 값은 **group**의 두 값으로 정확하게 지정하여 논리적으로 그룹화되어야 합니다. 로드 스크립트에서 표본 값에 대해 필드 이름을 지정하지 않으면 필드 이름으로 **Value**가 자동 지정됩니다.
grp	두 표본 그룹의 이름이 각각 포함된 필드입니다. 로드 스크립트에서 그룹에 대해 필드 이름을 지정하지 않으면 필드 이름으로 **Type**이 자동 지정됩니다.
eq_var	**eq_var**이 False(0)로 지정된 경우, 두 표본의 개별 분산으로 가정됩니다. **eq_var**이 True(1)로 지정된 경우, 표본 간의 등분산으로 가정됩니다.

제한 사항:

표현식 값에 텍스트 값, NULL 값, 누락된 값이 있으면 함수에서 NULL을 반환합니다.

```
TTest_sterr( Group, Value )
TTest_sterr( Group, Value, false )
```

관련 항목:

일반적인 t-test 보고서 만들기 (page 503)

TTest_t

TTest_t()는 두 독립 수열에 대해 집계된 t 값을 반환합니다.

이 함수는 독립 표본 학생 t 검정에 적용됩니다.

이 함수를 데이터 로드 스크립트에서 사용하면 group by 절로 정의된 여러 레코드에서 값이 반복됩니다.

차트 표현식에서 이 함수를 사용하면 차트 차원에 대해 값이 반복됩니다.

구문:

```
TTest_t(grp, value[, eq_var])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
value	평가할 표본 값입니다. 표본 값은 **group**의 두 값으로 정확하게 지정하여 논리적으로 그룹화되어야 합니다. 로드 스크립트에서 표본 값에 대해 필드 이름을 지정하지 않으면 필드 이름으로 **Value**가 자동 지정됩니다.
grp	두 표본 그룹의 이름이 각각 포함된 필드입니다. 로드 스크립트에서 그룹에 대해 필드 이름을 지정하지 않으면 필드 이름으로 **Type**이 자동 지정됩니다.
eq_var	**eq_var**이 False(0)로 지정된 경우, 두 표본의 개별 분산으로 가정됩니다. **eq_var**이 True(1)로 지정된 경우, 표본 간의 등분산으로 가정됩니다.

제한 사항:

표현식 값에 텍스트 값, NULL 값, 누락된 값이 있으면 함수에서 NULL을 반환합니다.

```
TTest_t( Group, Value, false )
```

관련 항목:

일반적인 t-test 보고서 만들기 (page 503)

TTest_upper

TTest_upper()는 두 독립 수열에 대한 신뢰 구간의 상단에 대해 집계된 값을 반환합니다.

이 함수는 독립 표본 학생 t 검정에 적용됩니다.

이 함수를 데이터 로드 스크립트에서 사용하면 group by 절로 정의된 여러 레코드에서 값이 반복됩니다.

차트 표현식에서 이 함수를 사용하면 차트 차원에 대해 값이 반복됩니다.

구문:

```
TTest_upper (grp, value [, sig [, eq_var]])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
value	평가할 표본 값입니다. 표본 값은 **group**의 두 값으로 정확하게 지정하여 논리적으로 그룹화되어야 합니다. 로드 스크립트에서 표본 값에 대해 필드 이름을 지정하지 않으면 필드 이름으로 **Value**가 자동 지정됩니다.
grp	두 표본 그룹의 이름이 각각 포함된 필드입니다. 로드 스크립트에서 그룹에 대해 필드 이름을 지정하지 않으면 필드 이름으로 **Type**이 자동 지정됩니다.
sig	양측 유의성 수준은 **sig**에 지정할 수 있습니다. 생략할 경우 **sig**는 0.025로 설정되며, 신뢰 구간은 95%가 됩니다.
eq_var	**eq_var**이 False(0)로 지정된 경우, 두 표본의 개별 분산으로 가정됩니다. **eq_var**이 True(1)로 지정된 경우, 표본 간의 등분산으로 가정됩니다.

제한 사항:

표현식 값에 텍스트 값, NULL 값, 누락된 값이 있으면 함수에서 NULL을 반환합니다.

```
TTest_upper( Group, Value )
TTest_upper( Group, Value, sig, false )
```

관련 항목:

일반적인 t-test 보고서 만들기 (page 503)

TTestw_conf

TTestw_conf()는 두 독립 수열에 대해 집계된 t 값을 반환합니다.

이 함수는 두 독립 표본 학생 t 검정에 적용되며, 여기서 입력 데이터 수열은 가중치가 적용된 2열 형식으로 지정됩니다.

이 함수를 데이터 로드 스크립트에서 사용하면 group by 절로 정의된 여러 레코드에서 값이 반복됩니다.

차트 표현식에서 이 함수를 사용하면 차트 차원에 대해 값이 반복됩니다.

구문:

```
TTestw_conf (weight, grp, value [, sig [, eq_var]])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
value	평가할 표본 값입니다. 표본 값은 **group**의 두 값으로 정확하게 지정하여 논리적으로 그룹화되어야 합니다. 로드 스크립트에서 표본 값에 대해 필드 이름을 지정하지 않으면 필드 이름으로 **Value**가 자동 지정됩니다.
weight	**value**의 각 값은 **weight**의 해당 가중치 값에 따라 한 번 이상 계수될 수 있습니다.
grp	두 표본 그룹의 이름이 각각 포함된 필드입니다. 로드 스크립트에서 그룹에 대해 필드 이름을 지정하지 않으면 필드 이름으로 **Type**이 자동 지정됩니다.
sig	양측 유의성 수준은 **sig**에 지정할 수 있습니다. 생략할 경우 **sig**는 0.025로 설정되며, 신뢰 구간은 95%가 됩니다.
eq_var	**eq_var**이 False(0)로 지정된 경우, 두 표본의 개별 분산으로 가정됩니다. **eq_var**이 True(1)로 지정된 경우, 표본 간의 등분산으로 가정됩니다.

제한 사항:

표현식 값에 텍스트 값, NULL 값, 누락된 값이 있으면 함수에서 NULL을 반환합니다.

```
TTestw_conf( Weight, Group, Value )
TTestw_conf( Weight, Group, Value, sig, false )
```

관련 항목:

일반적인 t-test 보고서 만들기 (page 503)

TTestw_df

TTestw_df()는 두 독립 수열에 대해 집계된 학생 t 검정 df 값(자유도)을 반환합니다.

이 함수는 두 독립 표본 학생 t 검정에 적용되며, 여기서 입력 데이터 수열은 가중치가 적용된 2열 형식으로 지정됩니다.

이 함수를 데이터 로드 스크립트에서 사용하면 group by 절로 정의된 여러 레코드에서 값이 반복됩니다.

차트 표현식에서 이 함수를 사용하면 차트 차원에 대해 값이 반복됩니다.

구문:

```
TTestw_df (weight, grp, value [, eq_var])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
weight	**value**의 각 값은 **weight**의 해당 가중치 값에 따라 한 번 이상 계수될 수 있습니다.
grp	두 표본 그룹의 이름이 각각 포함된 필드입니다. 로드 스크립트에서 그룹에 대해 필드 이름을 지정하지 않으면 필드 이름으로 **Type**이 자동 지정됩니다.
value	평가할 표본 값입니다. 표본 값은 **group**의 두 값으로 정확하게 지정하여 논리적으로 그룹화되어야 합니다. 로드 스크립트에서 표본 값에 대해 필드 이름을 지정하지 않으면 필드 이름으로 **Value**가 자동 지정됩니다.
eq_var	**eq_var**이 False(0)로 지정된 경우, 두 표본의 개별 분산으로 가정됩니다. **eq_var**이 True(1)로 지정된 경우, 표본 간의 등분산으로 가정됩니다.

제한 사항:

표현식 값에 텍스트 값, NULL 값, 누락된 값이 있으면 함수에서 NULL을 반환합니다.

```
TTestw_df( Weight, Group, Value )
TTestw_df( Weight, Group, Value, false )
```

관련 항목:

일반적인 t-test 보고서 만들기 (page 503)

TTestw_dif

TTestw_dif()는 두 독립 수열에 대해 집계된 학생 t 검정 평균값 차를 반환합니다.

이 함수는 두 독립 표본 학생 t 검정에 적용되며, 여기서 입력 데이터 수열은 가중치가 적용된 2열 형식으로 지정됩니다.

이 함수를 데이터 로드 스크립트에서 사용하면 group by 절로 정의된 여러 레코드에서 값이 반복됩니다.

차트 표현식에서 이 함수를 사용하면 차트 차원에 대해 값이 반복됩니다.

구문:

```
TTestw_dif (weight, grp, value)
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
weight	**value**의 각 값은 **weight**의 해당 가중치 값에 따라 한 번 이상 계수될 수 있습니다.
grp	두 표본 그룹의 이름이 각각 포함된 필드입니다. 로드 스크립트에서 그룹에 대해 필드 이름을 지정하지 않으면 필드 이름으로 **Type**이 자동 지정됩니다.
value	평가할 표본 값입니다. 표본 값은 **group**의 두 값으로 정확하게 지정하여 논리적으로 그룹화되어야 합니다. 로드 스크립트에서 표본 값에 대해 필드 이름을 지정하지 않으면 필드 이름으로 **Value**가 자동 지정됩니다.

제한 사항:

표현식 값에 텍스트 값, NULL 값, 누락된 값이 있으면 함수에서 NULL을 반환합니다.

```
TTestw_dif( Weight, Group, Value )
TTestw_dif( Weight, Group, Value, false )
```

관련 항목:

일반적인 t-test 보고서 만들기 (page 503)

TTestw_lower

TTestw_lower()는 두 독립 수열에 대한 신뢰 구간의 하단에 대해 집계된 값을 반환합니다.

이 함수는 두 독립 표본 학생 t 검정에 적용되며, 여기서 입력 데이터 수열은 가중치가 적용된 2열 형식으로 지정됩니다.

이 함수를 데이터 로드 스크립트에서 사용하면 group by 절로 정의된 여러 레코드에서 값이 반복됩니다.

차트 표현식에서 이 함수를 사용하면 차트 차원에 대해 값이 반복됩니다.

구문:

```
TTestw_lower (weight, grp, value [, sig [, eq_var]])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
weight	**value**의 각 값은 **weight**의 해당 가중치 값에 따라 한 번 이상 계수될 수 있습니다.
grp	두 표본 그룹의 이름이 각각 포함된 필드입니다. 로드 스크립트에서 그룹에 대해 필드 이름을 지정하지 않으면 필드 이름으로 **Type**이 자동 지정됩니다.
value	평가할 표본 값입니다. 표본 값은 **group**의 두 값으로 정확하게 지정하여 논리적으로 그룹화되어야 합니다. 로드 스크립트에서 표본 값에 대해 필드 이름을 지정하지 않으면 필드 이름으로 **Value**가 자동 지정됩니다.
sig	양측 유의성 수준은 **sig**에 지정할 수 있습니다. 생략할 경우 **sig**는 0.025로 설정되며, 신뢰 구간은 95%가 됩니다.
eq_var	**eq_var**이 False(0)로 지정된 경우, 두 표본의 개별 분산으로 가정됩니다. **eq_var**이 True(1)로 지정된 경우, 표본 간의 등분산으로 가정됩니다.

제한 사항:

표현식 값에 텍스트 값, NULL 값, 누락된 값이 있으면 함수에서 NULL을 반환합니다.

```
TTestw_lower( Weight, Group, Value )
TTestw_lower( Weight, Group, Value, sig, false )
```

관련 항목:

일반적인 t-test 보고서 만들기 (page 503)

TTestw_sig

TTestw_sig()는 두 독립 수열에 대해 집계된 학생 t 검정 양측 유의성 수준을 반환합니다.

이 함수는 두 독립 표본 학생 t 검정에 적용되며, 여기서 입력 데이터 수열은 가중치가 적용된 2열 형식으로 지정됩니다.

이 함수를 데이터 로드 스크립트에서 사용하면 group by 절로 정의된 여러 레코드에서 값이 반복됩니다.

차트 표현식에서 이 함수를 사용하면 차트 차원에 대해 값이 반복됩니다.

구문:

```
TTestw_sig ( weight, grp, value [, eq_var])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
weight	**value**의 각 값은 **weight**의 해당 가중치 값에 따라 한 번 이상 계수될 수 있습니다.
grp	두 표본 그룹의 이름이 각각 포함된 필드입니다. 로드 스크립트에서 그룹에 대해 필드 이름을 지정하지 않으면 필드 이름으로 **Type**이 자동 지정됩니다.
value	평가할 표본 값입니다. 표본 값은 **group**의 두 값으로 정확하게 지정하여 논리적으로 그룹화되어야 합니다. 로드 스크립트에서 표본 값에 대해 필드 이름을 지정하지 않으면 필드 이름으로 **Value**가 자동 지정됩니다.
eq_var	**eq_var**이 False(0)로 지정된 경우, 두 표본의 개별 분산으로 가정됩니다. **eq_var**이 True(1)로 지정된 경우, 표본 간의 등분산으로 가정됩니다.

제한 사항:

표현식 값에 텍스트 값, NULL 값, 누락된 값이 있으면 함수에서 NULL을 반환합니다.

```
TTestw_sig( Weight, Group, Value )
TTestw_sig( Weight, Group, Value, false )
```

관련 항목:

일반적인 t-test 보고서 만들기 (page 503)

TTestw_sterr

TTestw_sterr()은 두 독립 수열에 대해 집계된 학생 t 검정 평균값 차의 표준 오차를 반환합니다.

이 함수는 두 독립 표본 학생 t 검정에 적용되며, 여기서 입력 데이터 수열은 가중치가 적용된 2열 형식으로 지정됩니다.

이 함수를 데이터 로드 스크립트에서 사용하면 group by 절로 정의된 여러 레코드에서 값이 반복됩니다.

차트 표현식에서 이 함수를 사용하면 차트 차원에 대해 값이 반복됩니다.

구문:

```
TTestw_sterr (weight, grp, value [, eq_var])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
weight	**value**의 각 값은 **weight**의 해당 가중치 값에 따라 한 번 이상 계수될 수 있습니다.
grp	두 표본 그룹의 이름이 각각 포함된 필드입니다. 로드 스크립트에서 그룹에 대해 필드 이름을 지정하지 않으면 필드 이름으로 **Type**이 자동 지정됩니다.
value	평가할 표본 값입니다. 표본 값은 **group**의 두 값으로 정확하게 지정하여 논리적으로 그룹화되어야 합니다. 로드 스크립트에서 표본 값에 대해 필드 이름을 지정하지 않으면 필드 이름으로 **Value**가 자동 지정됩니다.
eq_var	**eq_var**이 False(0)로 지정된 경우, 두 표본의 개별 분산으로 가정됩니다. **eq_var**이 True(1)로 지정된 경우, 표본 간의 등분산으로 가정됩니다.

제한 사항:

표현식 값에 텍스트 값, NULL 값, 누락된 값이 있으면 함수에서 NULL을 반환합니다.

```
TTestw_sterr( Weight, Group, Value )
TTestw_sterr( Weight, Group, Value, false )
```

관련 항목:

일반적인 t-test 보고서 만들기 (page 503)

TTestw_t

TTestw_t()는 두 독립 수열에 대해 집계된 t 값을 반환합니다.

이 함수는 두 독립 표본 학생 t 검정에 적용되며, 여기서 입력 데이터 수열은 가중치가 적용된 2열 형식으로 지정됩니다.

이 함수를 데이터 로드 스크립트에서 사용하면 group by 절로 정의된 여러 레코드에서 값이 반복됩니다.

차트 표현식에서 이 함수를 사용하면 차트 차원에 대해 값이 반복됩니다.

구문:

```
ttestw_t (weight, grp, value [, eq_var])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
value	평가할 표본 값입니다. 표본 값은 **group**의 두 값으로 정확하게 지정하여 논리적으로 그룹화되어야 합니다. 로드 스크립트에서 표본 값에 대해 필드 이름을 지정하지 않으면 필드 이름으로 **Value**가 자동 지정됩니다.
weight	**value**의 각 값은 **weight**의 해당 가중치 값에 따라 한 번 이상 계수될 수 있습니다.
grp	두 표본 그룹의 이름이 각각 포함된 필드입니다. 로드 스크립트에서 그룹에 대해 필드 이름을 지정하지 않으면 필드 이름으로 **Type**이 자동 지정됩니다.
eq_var	**eq_var**이 False(0)로 지정된 경우, 두 표본의 개별 분산으로 가정됩니다. **eq_var**이 True(1)로 지정된 경우, 표본 간의 등분산으로 가정됩니다.

제한 사항:

표현식 값에 텍스트 값, NULL 값, 누락된 값이 있으면 함수에서 NULL을 반환합니다.

```
TTestw_t( Weight, Group, Value )
TTestw_t( Weight, Group, Value, false )
```

관련 항목:

일반적인 t-test 보고서 만들기 (page 503)

TTestw_upper

TTestw_upper()는 두 독립 수열에 대한 신뢰 구간의 상단에 대해 집계된 값을 반환합니다.

이 함수는 두 독립 표본 학생 t 검정에 적용되며, 여기서 입력 데이터 수열은 가중치가 적용된 2열 형식으로 지정됩니다.

이 함수를 데이터 로드 스크립트에서 사용하면 group by 절로 정의된 여러 레코드에서 값이 반복됩니다.

차트 표현식에서 이 함수를 사용하면 차트 차원에 대해 값이 반복됩니다.

구문:

```
TTestw_upper (weight, grp, value [, sig [, eq_var]])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
weight	**value**의 각 값은 **weight**의 해당 가중치 값에 따라 한 번 이상 계수될 수 있습니다.
grp	두 표본 그룹의 이름이 각각 포함된 필드입니다. 로드 스크립트에서 그룹에 대해 필드 이름을 지정하지 않으면 필드 이름으로 **Type**이 자동 지정됩니다.
value	평가할 표본 값입니다. 표본 값은 **group**의 두 값으로 정확하게 지정하여 논리적으로 그룹화되어야 합니다. 로드 스크립트에서 표본 값에 대해 필드 이름을 지정하지 않으면 필드 이름으로 **Value**가 자동 지정됩니다.
sig	양측 유의성 수준은 **sig**에 지정할 수 있습니다. 생략할 경우 **sig**는 0.025로 설정되며, 신뢰 구간은 95%가 됩니다.
eq_var	**eq_var**이 False(0)로 지정된 경우, 두 표본의 개별 분산으로 가정됩니다. **eq_var**이 True(1)로 지정된 경우, 표본 간의 등분산으로 가정됩니다.

제한 사항:

표현식 값에 텍스트 값, NULL 값, 누락된 값이 있으면 함수에서 NULL을 반환합니다.

```
TTestw_upper( Weight, Group, Value )
TTestw_upper( Weight, Group, Value, sig, false )
```

관련 항목:

일반적인 t-test 보고서 만들기 (page 503)

TTest1_conf

TTest1_conf()는 수열에 대해 집계된 신뢰 구간 값을 반환합니다.

이 함수는 단일 표본 학생 t 검정에 적용됩니다.

이 함수를 데이터 로드 스크립트에서 사용하면 group by 절로 정의된 여러 레코드에서 값이 반복됩니다.

차트 표현식에서 이 함수를 사용하면 차트 차원에 대해 값이 반복됩니다.

구문:

```
TTest1_conf (value [, sig ])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
value	평가할 표본입니다. 로드 스크립트에서 표본 값에 대해 필드 이름을 지정하지 않으면 필드 이름으로 **Value**가 자동 지정됩니다.
sig	양측 유의성 수준은 **sig**에 지정할 수 있습니다. 생략할 경우 **sig**는 0.025로 설정되며, 신뢰 구간은 95%가 됩니다.

제한 사항:

표현식 값에 텍스트 값, NULL 값, 누락된 값이 있으면 함수에서 NULL을 반환합니다.

```
TTest1_conf( value )
TTest1_conf( value, 0.005 )
```

관련 항목:

일반적인 t-test 보고서 만들기 (page 503)

TTest1_df

TTest1_df()는 수열에 대해 집계된 학생 t 검정 df 값(자유도)을 반환합니다.

이 함수는 단일 표본 학생 t 검정에 적용됩니다.

이 함수를 데이터 로드 스크립트에서 사용하면 group by 절로 정의된 여러 레코드에서 값이 반복됩니다.

차트 표현식에서 이 함수를 사용하면 차트 차원에 대해 값이 반복됩니다.

구문:

```
TTest1_df (value)
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
value	평가할 표본입니다. 로드 스크립트에서 표본 값에 대해 필드 이름을 지정하지 않으면 필드 이름으로 **Value**가 자동 지정됩니다.

제한 사항:

표현식 값에 텍스트 값, NULL 값, 누락된 값이 있으면 함수에서 NULL을 반환합니다.

```
TTest1_df( Value )
```

관련 항목:

일반적인 t-test 보고서 만들기 (page 503)

TTest1_dif

TTest1_dif()는 수열에 대해 집계된 학생 t 검정 평균값 차를 반환합니다.

이 함수는 단일 표본 학생 t 검정에 적용됩니다.

이 함수를 데이터 로드 스크립트에서 사용하면 group by 절로 정의된 여러 레코드에서 값이 반복됩니다.

차트 표현식에서 이 함수를 사용하면 차트 차원에 대해 값이 반복됩니다.

구문:

```
TTest1_dif (value)
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
value	평가할 표본입니다. 로드 스크립트에서 표본 값에 대해 필드 이름을 지정하지 않으면 필드 이름으로 **Value**가 자동 지정됩니다.

제한 사항:

표현식 값에 텍스트 값, NULL 값, 누락된 값이 있으면 함수에서 NULL을 반환합니다.

```
TTest1_dif( Value )
```

관련 항목:

일반적인 t-test 보고서 만들기 (page 503)

TTest1_lower

TTest1_lower()는 수열에 대한 신뢰 구간의 하단에 대해 집계된 값을 반환합니다.

이 함수는 단일 표본 학생 t 검정에 적용됩니다.

이 함수를 데이터 로드 스크립트에서 사용하면 group by 절로 정의된 여러 레코드에서 값이 반복됩니다.

차트 표현식에서 이 함수를 사용하면 차트 차원에 대해 값이 반복됩니다.

구문:

```
TTest1_lower (value [, sig])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
value	평가할 표본입니다. 로드 스크립트에서 표본 값에 대해 필드 이름을 지정하지 않으면 필드 이름으로 **Value**가 자동 지정됩니다.
sig	양측 유의성 수준은 **sig**에 지정할 수 있습니다. 생략할 경우 **sig**는 0.025로 설정되며, 신뢰 구간은 95%가 됩니다.

제한 사항:

표현식 값에 텍스트 값, NULL 값, 누락된 값이 있으면 함수에서 NULL을 반환합니다.

```
TTest1_lower( Value )
TTest1_lower( Value, 0.005 )
```

관련 항목:

일반적인 t-test 보고서 만들기 (page 503)

TTest1_sig

TTest1_sig()는 수열에 대해 집계된 학생 t 검정 양측 유의성 수준을 반환합니다.

이 함수는 단일 표본 학생 t 검정에 적용됩니다.

이 함수를 데이터 로드 스크립트에서 사용하면 group by 절로 정의된 여러 레코드에서 값이 반복됩니다.

차트 표현식에서 이 함수를 사용하면 차트 차원에 대해 값이 반복됩니다.

구문:

```
TTest1_sig (value)
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
value	평가할 표본입니다. 로드 스크립트에서 표본 값에 대해 필드 이름을 지정하지 않으면 필드 이름으로 **Value**가 자동 지정됩니다.

제한 사항:

표현식 값에 텍스트 값, NULL 값, 누락된 값이 있으면 함수에서 NULL을 반환합니다.

```
TTest1_sig( Value )
```

관련 항목:

일반적인 t-test 보고서 만들기 (page 503)

TTest1_sterr

TTest1_sterr()은 수열에 대해 집계된 학생 t 검정 평균값 차의 표준 오차를 반환합니다.

이 함수는 단일 표본 학생 t 검정에 적용됩니다.

이 함수를 데이터 로드 스크립트에서 사용하면 group by 절로 정의된 여러 레코드에서 값이 반복됩니다.

차트 표현식에서 이 함수를 사용하면 차트 차원에 대해 값이 반복됩니다.

구문:

```
TTest1_sterr (value)
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
value	평가할 표본입니다. 로드 스크립트에서 표본 값에 대해 필드 이름을 지정하지 않으면 필드 이름으로 **Value**가 자동 지정됩니다.

제한 사항:

표현식 값에 텍스트 값, NULL 값, 누락된 값이 있으면 함수에서 NULL을 반환합니다.

```
TTest1_sterr( Value )
```

관련 항목:

- *일반적인 t-test 보고서 만들기 (page 503)*

TTest1_t

TTest1_t() 는 수열에 대해 집계된 t 값을 반환합니다.

이 함수는 단일 표본 학생 t 검정에 적용됩니다.

이 함수를 데이터 로드 스크립트에서 사용하면 group by 절로 정의된 여러 레코드에서 값이 반복됩니다.

차트 표현식에서 이 함수를 사용하면 차트 차원에 대해 값이 반복됩니다.

구문:

```
TTest1_t (value)
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
value	평가할 표본입니다. 로드 스크립트에서 표본 값에 대해 필드 이름을 지정하지 않으면 필드 이름으로 **Value**가 자동 지정됩니다.

제한 사항:

표현식 값에 텍스트 값, NULL 값, 누락된 값이 있으면 함수에서 NULL을 반환합니다.

```
TTest1_t( Value )
```

관련 항목:

- *일반적인 t-test 보고서 만들기 (page 503)*

TTest1_upper

TTest1_upper()는 수열에 대한 신뢰 구간의 상단에 대해 집계된 값을 반환합니다.

이 함수는 단일 표본 학생 t 검정에 적용됩니다.

이 함수를 데이터 로드 스크립트에서 사용하면 group by 절로 정의된 여러 레코드에서 값이 반복됩니다.

차트 표현식에서 이 함수를 사용하면 차트 차원에 대해 값이 반복됩니다.

구문:

```
TTest1_upper (value [, sig])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
value	평가할 표본입니다. 로드 스크립트에서 표본 값에 대해 필드 이름을 지정하지 않으면 필드 이름으로 **Value**가 자동 지정됩니다.
sig	양측 유의성 수준은 **sig**에 지정할 수 있습니다. 생략할 경우 **sig**는 0.025로 설정되며, 신뢰 구간은 95%가 됩니다.

제한 사항:

표현식 값에 텍스트 값, NULL 값, 누락된 값이 있으면 함수에서 NULL을 반환합니다.

```
TTest1_upper( Value )
TTest1_upper( Value, 0.005 )
```

관련 항목:

일반적인 t-test 보고서 만들기 (page 503)

TTest1w_conf

TTest1w_conf()는 수열에 대해 집계된 신뢰 구간 값을 반환하는 **숫자** 함수입니다.

이 함수는 단일 표본 학생 t 검정에 적용되며, 여기서 입력 데이터 수열은 가중치가 적용된 2열 형식으로 지정됩니다.

이 함수를 데이터 로드 스크립트에서 사용하면 group by 절로 정의된 여러 레코드에서 값이 반복됩니다.

차트 표현식에서 이 함수를 사용하면 차트 차원에 대해 값이 반복됩니다.

구문:

```
TTest1w_conf (weight, value [, sig ])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
value	평가할 표본입니다. 로드 스크립트에서 표본 값에 대해 필드 이름을 지정하지 않으면 필드 이름으로 **Value**가 자동 지정됩니다.
weight	**value**의 각 값은 **weight**의 해당 가중치 값에 따라 한 번 이상 계수될 수 있습니다.
sig	양측 유의성 수준은 **sig**에 지정할 수 있습니다. 생략할 경우 **sig**는 0.025로 설정되며, 신뢰 구간은 95%가 됩니다.

제한 사항:

표현식 값에 텍스트 값, NULL 값, 누락된 값이 있으면 함수에서 NULL을 반환합니다.

```
TTest1w_conf( Weight, Value )
TTest1w_conf( Weight, Value, 0.005 )
```

관련 항목:

일반적인 t-test 보고서 만들기 (page 503)

TTest1w_df

TTest1w_df()는 수열에 대해 집계된 학생 t 검정 df 값(자유도)을 반환합니다.

이 함수는 단일 표본 학생 t 검정에 적용되며, 여기서 입력 데이터 수열은 가중치가 적용된 2열 형식으로 지정됩니다.

이 함수를 데이터 로드 스크립트에서 사용하면 group by 절로 정의된 여러 레코드에서 값이 반복됩니다.

차트 표현식에서 이 함수를 사용하면 차트 차원에 대해 값이 반복됩니다.

구문:

```
TTest1w_df (weight, value)
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
value	평가할 표본입니다. 로드 스크립트에서 표본 값에 대해 필드 이름을 지정하지 않으면 필드 이름으로 **Value**가 자동 지정됩니다.
weight	**value**의 각 값은 **weight**의 해당 가중치 값에 따라 한 번 이상 계수될 수 있습니다.

제한 사항:

표현식 값에 텍스트 값, NULL 값, 누락된 값이 있으면 함수에서 NULL을 반환합니다.

```
TTest1w_df( Weight, Value )
```

관련 항목:

일반적인 t-test 보고서 만들기 (page 503)

TTest1w_dif

TTest1w_dif()는 수열에 대해 집계된 학생 t 검정 평균값 차를 반환합니다.

이 함수는 단일 표본 학생 t 검정에 적용되며, 여기서 입력 데이터 수열은 가중치가 적용된 2열 형식으로 지정됩니다.

이 함수를 데이터 로드 스크립트에서 사용하면 group by 절로 정의된 여러 레코드에서 값이 반복됩니다.

차트 표현식에서 이 함수를 사용하면 차트 차원에 대해 값이 반복됩니다.

구문:

```
TTest1w_dif (weight, value)
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
value	평가할 표본입니다. 로드 스크립트에서 표본 값에 대해 필드 이름을 지정하지 않으면 필드 이름으로 **Value**가 자동 지정됩니다.
weight	**value**의 각 값은 **weight**의 해당 가중치 값에 따라 한 번 이상 계수될 수 있습니다.

제한 사항:

표현식 값에 텍스트 값, NULL 값, 누락된 값이 있으면 함수에서 NULL을 반환합니다.

```
TTest1w_dif( Weight, Value )
```

관련 항목:

일반적인 t-test 보고서 만들기 (page 503)

TTest1w_lower

TTest1w_lower()는 수열에 대한 신뢰 구간의 하단에 대해 집계된 값을 반환합니다.

이 함수는 단일 표본 학생 t 검정에 적용되며, 여기서 입력 데이터 수열은 가중치가 적용된 2열 형식으로 지정됩니다.

이 함수를 데이터 로드 스크립트에서 사용하면 group by 절로 정의된 여러 레코드에서 값이 반복됩니다.

차트 표현식에서 이 함수를 사용하면 차트 차원에 대해 값이 반복됩니다.

구문:

```
TTest1w_lower (weight, value [, sig ])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
value	평가할 표본입니다. 로드 스크립트에서 표본 값에 대해 필드 이름을 지정하지 않으면 필드 이름으로 **Value**가 자동 지정됩니다.
weight	**value**의 각 값은 **weight**의 해당 가중치 값에 따라 한 번 이상 계수될 수 있습니다.
sig	양측 유의성 수준은 **sig**에 지정할 수 있습니다. 생략할 경우 **sig**는 0.025로 설정되며, 신뢰 구간은 95%가 됩니다.

제한 사항:

표현식 값에 텍스트 값, NULL 값, 누락된 값이 있으면 함수에서 NULL을 반환합니다.

```
TTest1w_lower( Weight, Value )
TTest1w_lower( Weight, Value, 0.005 )
```

관련 항목:

일반적인 t-test 보고서 만들기 (page 503)

TTest1w_sig

TTest1w_sig()는 수열에 대해 집계된 학생 t 검정 양측 유의성 수준을 반환합니다.

이 함수는 단일 표본 학생 t 검정에 적용되며, 여기서 입력 데이터 수열은 가중치가 적용된 2열 형식으로 지정됩니다.

이 함수를 데이터 로드 스크립트에서 사용하면 group by 절로 정의된 여러 레코드에서 값이 반복됩니다.

차트 표현식에서 이 함수를 사용하면 차트 차원에 대해 값이 반복됩니다.

구문:

```
TTest1w_sig (weight, value)
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
value	평가할 표본입니다. 로드 스크립트에서 표본 값에 대해 필드 이름을 지정하지 않으면 필드 이름으로 **Value**가 자동 지정됩니다.
weight	**value**의 각 값은 **weight**의 해당 가중치 값에 따라 한 번 이상 계수될 수 있습니다.

제한 사항:

표현식 값에 텍스트 값, NULL 값, 누락된 값이 있으면 함수에서 NULL을 반환합니다.

```
TTest1w_sig( Weight, Value )
```

관련 항목:

일반적인 t-test 보고서 만들기 (page 503)

TTest1w_sterr

TTest1w_sterr()은 수열에 대해 집계된 학생 t 검정 평균값 차의 표준 오차를 반환합니다.

이 함수는 단일 표본 학생 t 검정에 적용되며, 여기서 입력 데이터 수열은 가중치가 적용된 2열 형식으로 지정됩니다.

이 함수를 데이터 로드 스크립트에서 사용하면 group by 절로 정의된 여러 레코드에서 값이 반복됩니다.

차트 표현식에서 이 함수를 사용하면 차트 차원에 대해 값이 반복됩니다.

구문:

```
TTest1w_sterr (weight, value)
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
value	평가할 표본입니다. 로드 스크립트에서 표본 값에 대해 필드 이름을 지정하지 않으면 필드 이름으로 **Value**가 자동 지정됩니다.
weight	**value**의 각 값은 **weight**의 해당 가중치 값에 따라 한 번 이상 계수될 수 있습니다.

제한 사항:

표현식 값에 텍스트 값, NULL 값, 누락된 값이 있으면 함수에서 NULL을 반환합니다.

```
TTest1w_sterr( Weight, Value )
```

관련 항목:

일반적인 t-test 보고서 만들기 (page 503)

TTest1w_t

TTest1w_t() 는 수열에 대해 집계된 t 값을 반환합니다.

이 함수는 단일 표본 학생 t 검정에 적용되며, 여기서 입력 데이터 수열은 가중치가 적용된 2열 형식으로 지정됩니다.

이 함수를 데이터 로드 스크립트에서 사용하면 group by 절로 정의된 여러 레코드에서 값이 반복됩니다.

차트 표현식에서 이 함수를 사용하면 차트 차원에 대해 값이 반복됩니다.

구문:

```
TTest1w_t ( weight, value)
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
value	평가할 표본입니다. 로드 스크립트에서 표본 값에 대해 필드 이름을 지정하지 않으면 필드 이름으로 **Value**가 자동 지정됩니다.
weight	**value**의 각 값은 **weight**의 해당 가중치 값에 따라 한 번 이상 계수될 수 있습니다.

제한 사항:

표현식 값에 텍스트 값, NULL 값, 누락된 값이 있으면 함수에서 NULL을 반환합니다.

```
TTest1w_t( Weight, Value )
```

관련 항목:

일반적인 t-test 보고서 만들기 (page 503)

TTest1w_upper

TTest1w_upper()는 수열에 대한 신뢰 구간의 상단에 대해 집계된 값을 반환합니다.

이 함수는 단일 표본 학생 t 검정에 적용되며, 여기서 입력 데이터 수열은 가중치가 적용된 2열 형식으로 지정됩니다.

이 함수를 데이터 로드 스크립트에서 사용하면 group by 절로 정의된 여러 레코드에서 값이 반복됩니다.

차트 표현식에서 이 함수를 사용하면 차트 차원에 대해 값이 반복됩니다.

구문:

```
TTest1w_upper (weight, value [, sig])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
value	평가할 표본입니다. 로드 스크립트에서 표본 값에 대해 필드 이름을 지정하지 않으면 필드 이름으로 **Value**가 자동 지정됩니다.
weight	**value**의 각 값은 **weight**의 해당 가중치 값에 따라 한 번 이상 계수될 수 있습니다.

인수	설명
sig	양측 유의성 수준은 **sig**에 지정할 수 있습니다. 생략할 경우 **sig**는 0.025로 설정되며, 신뢰 구간은 95%가 됩니다.

제한 사항:

표현식 값에 텍스트 값, NULL 값, 누락된 값이 있으면 함수에서 NULL을 반환합니다.

```
TTest1w_upper( Weight, Value )
TTest1w_upper( Weight, Value, 0.005 )
```

관련 항목:

일반적인 t-test 보고서 만들기 (page 503)

Z-test 함수

두 모평균의 통계 시험입니다. 두 표본 z 검정은 두 표본이 서로 다른지 시험하며 두 정규 분포가 알려진 분포일 경우 및 실험에 큰 크기의 표본을 사용할 경우에 주로 사용됩니다.

Z-test 통계 검정 함수는 함수에 적용되는 입력 데이터 수열의 유형에 따라 그룹화됩니다.

이 함수를 데이터 로드 스크립트에서 사용하면 group by 절로 정의된 여러 레코드에서 값이 반복됩니다.

차트 표현식에서 이 함수를 사용하면 차트 차원에 대해 값이 반복됩니다.

z-test 함수를 사용하는 방법의 예 (page 506)

단일 열 서식 함수

다음 함수가 단일 입력 데이터 수열이 포함된 z 검정에 적용됩니다.

ztest_conf
ZTest_conf()는 수열에 대해 집계된 z 값을 반환합니다.

`ZTest_conf()는 수열에 대해 집계된 z 값을 반환합니다. (value [, sigma [, sig ])`

ztest_dif
ZTest_dif()는 수열에 대해 집계된 z 검정 평균값 차를 반환합니다.

`ZTest_dif()는 수열에 대해 집계된 z 검정 평균값 차를 반환합니다. (value [, sigma])`

ztest_sig
ZTest_sig()는 수열에 대해 집계된 z 검정 양측 유의성 수준을 반환합니다.

`ZTest_sig()는 수열에 대해 집계된 z 검정 양측 유의성 수준을 반환합니다. (value [, sigma])`

ztest_sterr
ZTest_sterr()은 수열에 대해 집계된 z 검정 평균값 차의 표준 오차를 반환합니다.

ZTest_sterr()은 수열에 대해 집계된 z 검정 평균값 차의 표준 오차를 반환합니다. (value [, sigma])

ztest_z
ZTest_z()는 수열에 대해 집계된 z 값을 반환합니다.

ZTest_z()는 수열에 대해 집계된 z 값을 반환합니다. (value [, sigma])

ztest_lower
ZTest_lower()는 두 독립 수열에 대한 신뢰 구간의 하단에 대해 집계된 값을 반환합니다.

ZTest_lower()는 두 독립 수열에 대한 신뢰 구간의 하단에 대해 집계된 값을 반환합니다. (grp, value [, sig [, eq_var]])

ztest_upper
ZTest_upper()는 두 독립 수열에 대한 신뢰 구간의 상단에 대해 집계된 값을 반환합니다.

ZTest_upper()는 두 독립 수열에 대한 신뢰 구간의 상단에 대해 집계된 값을 반환합니다. (grp, value [, sig [, eq_var]])

가중 2열 서식 함수

다음 함수가 z 검정에 적용되며, 여기서 입력 데이터 수열은 가중치가 적용된 2열 형식으로 지정됩니다.

ztestw_conf
ZTestw_conf()는 수열에 대해 집계된 z 신뢰 구간 값을 반환합니다.

ZTestw_conf()는 수열에 대해 집계된 z 신뢰 구간 값을 반환합니다. (weight, value [, sigma [, sig]])

ztestw_dif
ZTestw_dif()는 수열에 대해 집계된 z 검정 평균값 차를 반환합니다.

ZTestw_dif()는 수열에 대해 집계된 z 검정 평균값 차를 반환합니다. (weight, value [, sigma])

ztestw_lower
ZTestw_lower()는 두 독립 수열에 대한 신뢰 구간의 하단에 대해 집계된 값을 반환합니다.

ZTestw_lower()는 두 독립 수열에 대한 신뢰 구간의 하단에 대해 집계된 값을 반환합니다. (weight, value [, sigma])

ztestw_sig
ZTestw_sig()는 수열에 대해 집계된 z 검정 양측 유의성 수준을 반환합니다.

ZTestw_sig()는 수열에 대해 집계된 z 검정 양측 유의성 수준을 반환합니다. (weight, value [, sigma])

ztestw_sterr
ZTestw_sterr()은 수열에 대해 집계된 z 검정 평균값 차의 표준 오차를 반환합니다.

ZTestw_sterr()은 수열에 대해 집계된 z 검정 평균값 차의 표준 오차를 반환합니다.
(weight, value [, sigma]**)**

ztestw_upper

ZTestw_upper()는 두 독립 수열에 대한 신뢰 구간의 상단에 대해 집계된 값을 반환합니다.

ZTestw_upper()는 두 독립 수열에 대한 신뢰 구간의 상단에 대해 집계된 값을 반환합니다.
(weight, value [, sigma]**)**

ztestw_z

ZTestw_z()는 수열에 대해 집계된 z 값을 반환합니다.

ZTestw_z()는 수열에 대해 집계된 z 값을 반환합니다. (weight, value [, sigma]**)**

ZTest_z

ZTest_z()는 수열에 대해 집계된 z 값을 반환합니다.

이 함수를 데이터 로드 스크립트에서 사용하면 group by 절로 정의된 여러 레코드에서 값이 반복됩니다.

차트 표현식에서 이 함수를 사용하면 차트 차원에 대해 값이 반복됩니다.

구문:

ZTest_z(value[, sigma]**)**

반환 데이터 유형: 숫자

인수:

인수

인수	설명
value	평가할 표본 값입니다. 모평균은 0으로 가정됩니다. 다른 평균에 대해 테스트를 수행하고자 하는 경우 표본 값에서 해당 평균값을 빼십시오.
sigma	표준 편차를 알고 있는 경우 **sigma**에 지정할 수 있습니다. **sigma**를 생략하면 실제 표본 표준 편차가 사용됩니다.

제한 사항:

표현식 값에 텍스트 값, NULL 값, 누락된 값이 있으면 함수에서 NULL을 반환합니다.

```
ZTest_z( Value-TestValue )
```

관련 항목:

- *z-test 함수를 사용하는 방법의 예 (page 506)*

ZTest_sig

ZTest_sig()는 수열에 대해 집계된 z 검정 양측 유의성 수준을 반환합니다.

이 함수를 데이터 로드 스크립트에서 사용하면 group by 절로 정의된 여러 레코드에서 값이 반복됩니다.

차트 표현식에서 이 함수를 사용하면 차트 차원에 대해 값이 반복됩니다.

구문:

```
ZTest_sig(value[, sigma])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
value	평가할 표본 값입니다. 모평균은 0으로 가정됩니다. 다른 평균에 대해 테스트를 수행하고자 하는 경우 표본 값에서 해당 평균값을 빼십시오.
sigma	표준 편차를 알고 있는 경우 **sigma**에 지정할 수 있습니다. **sigma**를 생략하면 실제 표본 표준 편차가 사용됩니다.

제한 사항:

표현식 값에 텍스트 값, NULL 값, 누락된 값이 있으면 함수에서 NULL을 반환합니다.

```
ZTest_sig(Value-TestValue)
```

관련 항목:

z-test 함수를 사용하는 방법의 예 (page 506)

ZTest_dif

ZTest_dif()는 수열에 대해 집계된 z 검정 평균값 차를 반환합니다.

이 함수를 데이터 로드 스크립트에서 사용하면 group by 절로 정의된 여러 레코드에서 값이 반복됩니다.

차트 표현식에서 이 함수를 사용하면 차트 차원에 대해 값이 반복됩니다.

구문:

```
ZTest_dif(value[, sigma])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
value	평가할 표본 값입니다. 모평균은 0으로 가정됩니다. 다른 평균에 대해 테스트를 수행하고자 하는 경우 표본 값에서 해당 평균값을 빼십시오.
sigma	표준 편차를 알고 있는 경우 **sigma**에 지정할 수 있습니다. **sigma**를 생략하면 실제 표본 표준 편차가 사용됩니다.

제한 사항:

표현식 값에 텍스트 값, NULL 값, 누락된 값이 있으면 함수에서 NULL을 반환합니다.

```
ZTest_dif(Value-TestValue)
```

관련 항목:

z-test 함수를 사용하는 방법의 예 (page 506)

ZTest_sterr

ZTest_sterr()은 수열에 대해 집계된 z 검정 평균값 차의 표준 오차를 반환합니다.

이 함수를 데이터 로드 스크립트에서 사용하면 group by 절로 정의된 여러 레코드에서 값이 반복됩니다.

차트 표현식에서 이 함수를 사용하면 차트 차원에 대해 값이 반복됩니다.

구문:

```
ZTest_sterr(value[, sigma])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
value	평가할 표본 값입니다. 모평균은 0으로 가정됩니다. 다른 평균에 대해 테스트를 수행하고자 하는 경우 표본 값에서 해당 평균값을 빼십시오.
sigma	표준 편차를 알고 있는 경우 **sigma**에 지정할 수 있습니다. **sigma**를 생략하면 실제 표본 표준 편차가 사용됩니다.

제한 사항:

표현식 값에 텍스트 값, NULL 값, 누락된 값이 있으면 함수에서 NULL을 반환합니다.

```
ZTest_sterr(Value-TestValue)
```

관련 항목:

z-test 함수를 사용하는 방법의 예 (page 506)

ZTest_conf

ZTest_conf()는 수열에 대해 집계된 z 값을 반환합니다.

이 함수를 데이터 로드 스크립트에서 사용하면 group by 절로 정의된 여러 레코드에서 값이 반복됩니다.

차트 표현식에서 이 함수를 사용하면 차트 차원에 대해 값이 반복됩니다.

구문:

```
ZTest_conf(value[, sigma[, sig]])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
value	평가할 표본 값입니다. 모평균은 0으로 가정됩니다. 다른 평균에 대해 테스트를 수행하고자 하는 경우 표본 값에서 해당 평균값을 빼십시오.
sigma	표준 편차를 알고 있는 경우 **sigma**에 지정할 수 있습니다. **sigma**를 생략하면 실제 표본 표준 편차가 사용됩니다.
sig	양측 유의성 수준은 **sig**에 지정할 수 있습니다. 생략할 경우 **sig**는 0.025로 설정되며, 신뢰 구간은 95%가 됩니다.

제한 사항:

표현식 값에 텍스트 값, NULL 값, 누락된 값이 있으면 함수에서 NULL을 반환합니다.

```
ZTest_conf(Value-TestValue)
```

관련 항목:

z-test 함수를 사용하는 방법의 예 (page 506)

ZTest_lower

ZTest_lower()는 두 독립 수열에 대한 신뢰 구간의 하단에 대해 집계된 값을 반환합니다.

이 함수를 데이터 로드 스크립트에서 사용하면 group by 절로 정의된 여러 레코드에서 값이 반복됩니다.

차트 표현식에서 이 함수를 사용하면 차트 차원에 대해 값이 반복됩니다.

구문:

```
ZTest_lower (grp, value [, sig [, eq_var]])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
value	평가할 표본 값입니다. 표본 값은 **group**의 두 값으로 정확하게 지정하여 논리적으로 그룹화되어야 합니다. 로드 스크립트에서 표본 값에 대해 필드 이름을 지정하지 않으면 필드 이름으로 **Value**가 자동 지정됩니다.
grp	두 표본 그룹의 이름이 각각 포함된 필드입니다. 로드 스크립트에서 그룹에 대해 필드 이름을 지정하지 않으면 필드 이름으로 **Type**이 자동 지정됩니다.
sig	양측 유의성 수준은 **sig**에 지정할 수 있습니다. 생략할 경우 **sig**는 0.025로 설정되며, 신뢰 구간은 95%가 됩니다.
eq_var	**eq_var**이 False(0)로 지정된 경우, 두 표본의 개별 분산으로 가정됩니다. **eq_var**이 True(1)로 지정된 경우, 표본 간의 등분산으로 가정됩니다.

제한 사항:

표현식 값에 텍스트 값, NULL 값, 누락된 값이 있으면 함수에서 NULL을 반환합니다.

```
ZTest_lower( Group, Value )
ZTest_lower( Group, Value, sig, false )
```

관련 항목:

z-test 함수를 사용하는 방법의 예 (page 506)

ZTest_upper

ZTest_upper()는 두 독립 수열에 대한 신뢰 구간의 상단에 대해 집계된 값을 반환합니다.

이 함수는 독립 표본 학생 t 검정에 적용됩니다.

이 함수를 데이터 로드 스크립트에서 사용하면 group by 절로 정의된 여러 레코드에서 값이 반복됩니다.

차트 표현식에서 이 함수를 사용하면 차트 차원에 대해 값이 반복됩니다.

구문:

```
ZTest_upper (grp, value [, sig [, eq_var]])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
value	평가할 표본 값입니다. 표본 값은 **group**의 두 값으로 정확하게 지정하여 논리적으로 그룹화되어야 합니다. 로드 스크립트에서 표본 값에 대해 필드 이름을 지정하지 않으면 필드 이름으로 **Value**가 자동 지정됩니다.
grp	두 표본 그룹의 이름이 각각 포함된 필드입니다. 로드 스크립트에서 그룹에 대해 필드 이름을 지정하지 않으면 필드 이름으로 **Type**이 자동 지정됩니다.
sig	양측 유의성 수준은 **sig**에 지정할 수 있습니다. 생략할 경우 **sig**는 0.025로 설정되며, 신뢰 구간은 95%가 됩니다.
eq_var	**eq_var**이 False(0)로 지정된 경우, 두 표본의 개별 분산으로 가정됩니다. **eq_var**이 True(1)로 지정된 경우, 표본 간의 등분산으로 가정됩니다.

제한 사항:

표현식 값에 텍스트 값, NULL 값, 누락된 값이 있으면 함수에서 NULL을 반환합니다.

```
ZTest_upper( Group, Value )
ZTest_upper( Group, Value, sig, false )
```

관련 항목:

z-test 함수를 사용하는 방법의 예 (page 506)

ZTestw_z

ZTestw_z()는 수열에 대해 집계된 z 값을 반환합니다.

이 함수는 z 검정에 적용되며, 여기서 입력 데이터 수열은 가중치가 적용된 2열 형식으로 지정됩니다.

이 함수를 데이터 로드 스크립트에서 사용하면 group by 절로 정의된 여러 레코드에서 값이 반복됩니다.

차트 표현식에서 이 함수를 사용하면 차트 차원에 대해 값이 반복됩니다.

구문:

```
ZTestw_z (weight, value [, sigma])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
value	값은 **value**에 의해 반환되어야 합니다. 표본 평균은 0으로 가정됩니다. 다른 평균에 대해 테스트를 수행하고자 하는 경우 표본 값에서 해당 값을 빼십시오.
weight	**value**의 각 표본 값은 **weight**의 해당 가중치 값에 따라 한 번 이상 계수될 수 있습니다.
sigma	표준 편차를 알고 있는 경우 **sigma**에 지정할 수 있습니다. **sigma**를 생략하면 실제 표본 표준 편차가 사용됩니다.

제한 사항:

표현식 값에 텍스트 값, NULL 값, 누락된 값이 있으면 함수에서 NULL을 반환합니다.

```
ZTestw_z( Weight, Value-TestValue)
```

관련 항목:

z-test 함수를 사용하는 방법의 예 (page 506)

ZTestw_sig

ZTestw_sig()는 수열에 대해 집계된 z 검정 양측 유의성 수준을 반환합니다.

이 함수는 z 검정에 적용되며, 여기서 입력 데이터 수열은 가중치가 적용된 2열 형식으로 지정됩니다.

이 함수를 데이터 로드 스크립트에서 사용하면 group by 절로 정의된 여러 레코드에서 값이 반복됩니다.

차트 표현식에서 이 함수를 사용하면 차트 차원에 대해 값이 반복됩니다.

구문:

```
ZTestw_sig (weight, value [, sigma])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
value	값은 **value**에 의해 반환되어야 합니다. 표본 평균은 0으로 가정됩니다. 다른 평균에 대해 테스트를 수행하고자 하는 경우 표본 값에서 해당 값을 빼십시오.

인수	설명
weight	**value**의 각 표본 값은 **weight**의 해당 가중치 값에 따라 한 번 이상 계수될 수 있습니다.
sigma	표준 편차를 알고 있는 경우 **sigma**에 지정할 수 있습니다. **sigma**를 생략하면 실제 표본 표준 편차가 사용됩니다.

제한 사항:

표현식 값에 텍스트 값, NULL 값, 누락된 값이 있으면 함수에서 NULL을 반환합니다.

```
ZTestw_sig( Weight, Value-TestValue)
```

관련 항목:

z-test 함수를 사용하는 방법의 예 (page 506)

ZTestw_dif

ZTestw_dif()는 수열에 대해 집계된 z 검정 평균값 차를 반환합니다.

이 함수는 z 검정에 적용되며, 여기서 입력 데이터 수열은 가중치가 적용된 2열 형식으로 지정됩니다.

이 함수를 데이터 로드 스크립트에서 사용하면 group by 절로 정의된 여러 레코드에서 값이 반복됩니다.

차트 표현식에서 이 함수를 사용하면 차트 차원에 대해 값이 반복됩니다.

구문:

```
ZTestw_dif ( weight, value [, sigma])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
value	값은 **value**에 의해 반환되어야 합니다. 표본 평균은 0으로 가정됩니다. 다른 평균에 대해 테스트를 수행하고자 하는 경우 표본 값에서 해당 값을 빼십시오.
weight	**value**의 각 표본 값은 **weight**의 해당 가중치 값에 따라 한 번 이상 계수될 수 있습니다.
sigma	표준 편차를 알고 있는 경우 **sigma**에 지정할 수 있습니다. **sigma**를 생략하면 실제 표본 표준 편차가 사용됩니다.

제한 사항:

표현식 값에 텍스트 값, NULL 값, 누락된 값이 있으면 함수에서 NULL을 반환합니다.

```
ZTestw_dif( Weight, Value-TestValue)
```

관련 항목:

z-test 함수를 사용하는 방법의 예 (page 506)

ZTestw_sterr

ZTestw_sterr()은 수열에 대해 집계된 z 검정 평균값 차의 표준 오차를 반환합니다.

이 함수는 z 검정에 적용되며, 여기서 입력 데이터 수열은 가중치가 적용된 2열 형식으로 지정됩니다.

이 함수를 데이터 로드 스크립트에서 사용하면 group by 절로 정의된 여러 레코드에서 값이 반복됩니다.

차트 표현식에서 이 함수를 사용하면 차트 차원에 대해 값이 반복됩니다.

구문:

```
ZTestw_sterr (weight, value [, sigma])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
value	값은 **value**에 의해 반환되어야 합니다. 표본 평균은 0으로 가정됩니다. 다른 평균에 대해 테스트를 수행하고자 하는 경우 표본 값에서 해당 값을 빼십시오.
weight	**value**의 각 표본 값은 **weight**의 해당 가중치 값에 따라 한 번 이상 계수될 수 있습니다.
sigma	표준 편차를 알고 있는 경우 **sigma**에 지정할 수 있습니다. **sigma**를 생략하면 실제 표본 표준 편차가 사용됩니다.

제한 사항:

표현식 값에 텍스트 값, NULL 값, 누락된 값이 있으면 함수에서 NULL을 반환합니다.

```
ZTestw_sterr( Weight, Value-TestValue)
```

관련 항목:

z-test 함수를 사용하는 방법의 예 (page 506)

ZTestw_conf

ZTestw_conf()는 수열에 대해 집계된 z 신뢰 구간 값을 반환합니다.

이 함수는 z 검정에 적용되며, 여기서 입력 데이터 수열은 가중치가 적용된 2열 형식으로 지정됩니다.

이 함수를 데이터 로드 스크립트에서 사용하면 group by 절로 정의된 여러 레코드에서 값이 반복됩니다.

차트 표현식에서 이 함수를 사용하면 차트 차원에 대해 값이 반복됩니다.

구문:

```
ZTest_conf(weight, value[, sigma[, sig]])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
value	평가할 표본 값입니다. 모평균은 0으로 가정됩니다. 다른 평균에 대해 테스트를 수행하고자 하는 경우 표본 값에서 해당 평균값을 빼십시오.
weight	**value**의 각 표본 값은 **weight**의 해당 가중치 값에 따라 한 번 이상 계수될 수 있습니다.
sigma	표준 편차를 알고 있는 경우 **sigma**에 지정할 수 있습니다. **sigma**를 생략하면 실제 표본 표준 편차가 사용됩니다.
sig	양측 유의성 수준은 **sig**에 지정할 수 있습니다. 생략할 경우 **sig**는 0.025로 설정되며, 신뢰 구간은 95%가 됩니다.

제한 사항:

표현식 값에 텍스트 값, NULL 값, 누락된 값이 있으면 함수에서 NULL을 반환합니다.

```
ZTestw_conf( Weight, Value-TestValue)
```

관련 항목:

z-test 함수를 사용하는 방법의 예 (page 506)

ZTestw_lower

ZTestw_lower()는 두 독립 수열에 대한 신뢰 구간의 하단에 대해 집계된 값을 반환합니다.

이 함수를 데이터 로드 스크립트에서 사용하면 group by 절로 정의된 여러 레코드에서 값이 반복됩니다.

차트 표현식에서 이 함수를 사용하면 차트 차원에 대해 값이 반복됩니다.

구문:

```
ZTestw_lower (grp, value [, sig [, eq_var]])
```

반환 데이터 유형: 숫자

인수:

인수	설명
value	평가할 표본 값입니다. 표본 값은 **group**의 두 값으로 정확하게 지정하여 논리적으로 그룹화되어야 합니다. 로드 스크립트에서 표본 값에 대해 필드 이름을 지정하지 않으면 필드 이름으로 **Value**가 자동 지정됩니다.
grp	두 표본 그룹의 이름이 각각 포함된 필드입니다. 로드 스크립트에서 그룹에 대해 필드 이름을 지정하지 않으면 필드 이름으로 **Type**이 자동 지정됩니다.
sig	양측 유의성 수준은 **sig**에 지정할 수 있습니다. 생략할 경우 **sig**는 0.025로 설정되며, 신뢰 구간은 95%가 됩니다.
eq_var	**eq_var**이 False(0)로 지정된 경우, 두 표본의 개별 분산으로 가정됩니다. **eq_var**이 True(1)로 지정된 경우, 표본 간의 등분산으로 가정됩니다.

제한 사항:

표현식 값에 텍스트 값, NULL 값, 누락된 값이 있으면 함수에서 NULL을 반환합니다.

```
ZTestw_lower( Group, Value )
ZTestw_lower( Group, Value, sig, false )
```

관련 항목:

z-test 함수를 사용하는 방법의 예 (page 506)

ZTestw_upper

ZTestw_upper()는 두 독립 수열에 대한 신뢰 구간의 상단에 대해 집계된 값을 반환합니다.

이 함수는 독립 표본 학생 t 검정에 적용됩니다.

이 함수를 데이터 로드 스크립트에서 사용하면 group by 절로 정의된 여러 레코드에서 값이 반복됩니다.

차트 표현식에서 이 함수를 사용하면 차트 차원에 대해 값이 반복됩니다.

구문:

```
ZTestw_upper (grp, value [, sig [, eq_var]])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
value	평가할 표본 값입니다. 표본 값은 **group**의 두 값으로 정확하게 지정하여 논리적으로 그룹화되어야 합니다. 로드 스크립트에서 표본 값에 대해 필드 이름을 지정하지 않으면 필드 이름으로 **Value**가 자동 지정됩니다.
grp	두 표본 그룹의 이름이 각각 포함된 필드입니다. 로드 스크립트에서 그룹에 대해 필드 이름을 지정하지 않으면 필드 이름으로 **Type**이 자동 지정됩니다.
sig	양측 유의성 수준은 **sig**에 지정할 수 있습니다. 생략할 경우 **sig**는 0.025로 설정되며, 신뢰 구간은 95%가 됩니다.
eq_var	**eq_var**이 False(0)로 지정된 경우, 두 표본의 개별 분산으로 가정됩니다. **eq_var**이 True(1)로 지정된 경우, 표본 간의 등분산으로 가정됩니다.

제한 사항:

표현식 값에 텍스트 값, NULL 값, 누락된 값이 있으면 함수에서 NULL을 반환합니다.

```
ZTestw_upper( Group, Value )
ZTestw_upper( Group, Value, sig, false )
```

관련 항목:

z-test 함수를 사용하는 방법의 예 (page 506)

통계 검정 함수의 예

이 섹션에는 차트 및 데이터 로드 스크립트에 적용되는 통계 검정 함수의 예가 나와 있습니다.

차트에 chi2-test 함수를 사용하는 방법의 예

chi2-test 함수는 chi 제곱 통계 분석과 관련된 값을 찾는 데 사용됩니다.

이 섹션에서는 샘플 데이터를 사용하여 시각화를 작성하고 Qlik Sense에서 사용 가능한 chi 제곱 분포 검정 함수의 값을 찾는 방법을 설명합니다. 구문과 인수에 대한 설명은 각 chi2-test 차트 함수 항목을 참조하십시오.

샘플에 사용할 데이터 로딩

스크립트에 로드될 세 가지 다른 통계 샘플을 설명하는 세 샘플 데이터 집합이 있습니다.

다음과 같이 하십시오.

1. 새 앱을 만듭니다.
   데이터 로드에서 다음을 입력합니다.
   ```
   // 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. 아이콘 을 클릭하여 데이터를 로드합니다.

chi2-test 차트 함수 시각화 만들기

샘플 1

다음과 같이 하십시오.

1. 데이터 로드 편집기에서 를 클릭해서 앱 보기로 이동한 후 이전에 만든 시트를 클릭합니다.
   시트 보기가 열립니다.
2. **시트 편집**을 클릭하여 시트를 편집합니다.
3. **차트**에서 테이블을 추가하고 **필드**에서 Grp, Grade 및 Count를 차원으로 추가합니다.
   이 테이블에 샘플 데이터가 표시됩니다.
4. 다음 표현식을 차원으로 갖는 다른 테이블을 추가합니다.
   `ValueList('p','df','Chi2')`
   이는 가상 차원 함수를 사용하여 세 가지 chi2-test 함수의 이름이 포함된 차원에 대한 레이블을 생성하는 것입니다.
   테이블에 다음 표현식을 측정값으로 추가합니다.
   `IF(ValueList('p','df','Chi2')='p',Chi2Test_p(Grp,Grade,Count),`
5. `IF(ValueList('p','df','Chi2')='df',Chi2Test_df(Grp,Grade,Count),`

   `Chi2Test_Chi2(Grp,Grade,Count)))`
   이에 따라 테이블에서 각 chi2-test 함수의 결과 값이 연결된 가상 차원 옆에 배치됩니다.
6. 측정값의 **숫자 서식**을 **숫자** 및 **유효숫자 3**개로 설정합니다.

측정값에 대한 표현식에 표현식을 대신 사용할 수도 있습니다. `Pick(Match(ValueList('p','df','Chi2'),'p','df','Chi2'),Chi2Test_p(Grp,Grade,Count),Chi2Test_df(Grp,Grade,Count),Chi2Test_Chi2(Grp,Grade,Count))`

결과:

샘플 1 데이터에 해당하는 chi2-test 함수의 결과 테이블에 다음과 같은 값이 포함됩니다.

결과 테이블

p	df	Chi2
0.820	5	2.21

샘플 2

다음과 같이 하십시오.

1. 예제 샘플 1에서 편집한 시트에서, **차트**에서는 테이블을 추가하고 **필드**에서는 Sex, Opinion 및 OpCount를 차원으로 추가합니다.
2. **복사** 및 **붙여넣기** 명령을 사용하여 샘플 1에서 결과 테이블의 복사본을 만듭니다. 측정값의 표현식을 편집하고 세 chi2-test 함수 모두의 인수를 샘플 2 데이터에 사용된 필드의 이름으로 대체합니다 (예: `Chi2Test_p(Sex,Opinion,OpCount)`).

결과:

샘플 2 데이터에 해당하는 chi2-test 함수의 결과 테이블에 다음과 같은 값이 포함됩니다.

결과 테이블

p	df	Chi2
0.000309	2	16.2

샘플 3

다음과 같이 하십시오.

1. 샘플 1 및 샘플 2 데이터에 대해 예제와 같이 두 개의 테이블을 더 만듭니다. 차원 테이블에서 차원으로 사용하는 필드는 Gender, Description, Actual 및 Expected입니다.
2. 결과 테이블에서 샘플 3 데이터에 사용된 필드의 이름을 사용합니다(예: `Chi2Test_p (Gender,Description,Actual,Expected)`).

결과:

샘플 3 데이터에 해당하는 chi2-test 함수의 결과 테이블에 다음과 같은 값이 포함됩니다.

결과 테이블

p	df	Chi2
0.000308	2	16.2

데이터 로드 스크립트에서 chi2-test 함수를 사용하는 방법의 예

chi2-test 함수는 chi 제곱 통계 분석과 관련된 값을 찾는 데 사용됩니다. 이 섹션에서는 Qlik Sense에서 사용 가능한 chi 제곱 분포 검정 함수를 데이터 로드 스크립트에서 사용하는 방법을 설명합니다. 구문과 인수에 대한 설명은 각 chi2-test 스크립트 함수 항목을 참조하십시오.

이 예에서는 두 그룹의 학생(I 및 II)에 대해 A-F의 성적을 받은 학생 수가 포함된 테이블을 사용합니다.

Data table

Group	A	B	C	D	E	F
I	15	7	9	20	26	19
II	10	11	7	15	21	16

샘플 데이터 로딩

다음과 같이 하십시오.

1. 새 앱을 만듭니다.
   데이터 로드 편집기에서 다음을 입력합니다.
   ```
   // 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. 아이콘 을 클릭하여 데이터를 로드합니다.

이제 샘플 데이터가 로드되었습니다.

chi2-test 함수 값 로드

이제 새 테이블의 샘플 데이터를 기준으로 Grp에 따라 그룹화된 chi2-test 값이 로드됩니다.

다음과 같이 하십시오.

1. 데이터 로드 편집기에서 스크립트의 끝 부분에 다음을 추가합니다.

```
// 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. 아이콘  을 클릭하여 데이터를 로드합니다.

이제 Chi2_table 테이블에 chi2-test 값이 로드되었습니다.

결과

다음과 같은 결과 chi2-test 값을 데이터 모델 뷰어의 **미리 보기** 아래에서 볼 수 있습니다.

Results

Grp	chi2	df	p
I	16.00	5	0.007
II	9.40	5	0.094

일반적인 t-test 보고서 만들기

일반적인 학생 t-test 보고서에는 **Group Statistics** 및 **Independent Samples Test** 결과가 있는 테이블이 포함될 수 있습니다.

다음 섹션에서는 두 개별 샘플 그룹 Observation 및 Comparison에 적용되는 Qlik Senset-test 함수를 사용하여 이러한 테이블을 만들겠습니다. 이 샘플에 해당하는 테이블은 다음과 같습니다.

그룹 통계

Type	N	Mean	Standard Deviation	Standard Error Mean
Comparison	20	11.95	14.61245	3.2674431
Observation	20	27.15	12.507997	2.7968933

독립 샘플 테스트

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

샘플 데이터 로딩

다음과 같이 하십시오.

1. 새 시트로 새 앱을 만듭니다.
2. 데이터 로드 편집기에서 다음을 입력합니다.
```
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 '|');
```
이 로드 스크립트에서는 **crosstable**에 3개의 인수가 필요하기 때문에 **recno()**가 포함됩니다. 따라서 **recno()**는 간편하게 추가 인수를 제공합니다(이 경우, 각 행의 ID). 생략하는 경우, **Comparison** 샘플 값이 로드되지 않습니다.
3. 아이콘 을 클릭하여 데이터를 로드합니다.

Group statistics 테이블 만들기

다음과 같이 하십시오.

1. 데이터 로드 편집기에서 를 클릭해서 앱 보기로 이동한 후 이전에 만든 시트를 클릭합니다. 그러면 시트 뷰가 열립니다.
2. **시트 편집**을 클릭하여 시트를 편집합니다.
3. **차트**에서 표를 추가하고 **필드**에서 Type을 차원으로 표에 추가합니다.
4. 다음 표현식을 측정값으로 추가합니다.

표현식 예

레이블	표현식
N	Count(Value)
Mean	Avg(Value)
Standard Deviation	Stdev(Value)
Standard Error Mean	Sterr(Value)

5. **정렬**을 클릭하고 Type이 정렬 목록 맨 위에 있는지 확인합니다.

결과:

이 샘플에 해당하는 Group statistics 테이블은 다음과 같습니다.

그룹 통계

Type	N	Mean	Standard Deviation	Standard Error Mean
Comparison	20	11.95	14.61245	3.2674431
Observation	20	27.15	12.507997	2.7968933

Independent sample test 테이블 만들기

다음과 같이 하십시오.

1. **시트 편집**을 클릭하여 시트를 편집합니다.
2. **차트**에서 다음 표현식이 포함된 표를 차원으로 표에 추가합니다. `=ValueList (Dual('Equal Variance not Assumed', 0), Dual('Equal Variance Assumed', 1))` 레이블을 Type으로 지정합니다.
3. 다음 표현식을 측정값으로 추가합니다.

표현식 예

레이블	표현식
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))

결과:

독립 샘플 테스트

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

z-test 함수를 사용하는 방법의 예

z-test 함수는 일반적으로 30개 이상의 대량 데이터 샘플에서 분산이 알려진 경우에 z-test 통계 분석과 관련된 값을 찾는 데 사용됩니다.

이 섹션에서는 샘플 데이터를 사용하여 시각화를 작성하고 z-test에서 사용 가능한 Qlik Sense 함수의 값을 찾는 방법을 설명합니다. 구문과 인수에 대한 설명은 각 z-test 차트 함수 항목을 참조하십시오.

샘플 데이터 로딩

여기에 사용되는 샘플 데이터는 t-test 함수 예제에 사용되는 것과 동일합니다. 샘플 데이터 크기는 일반적으로 z-test 분석에 너무 작은 것으로 간주되지만, Qlik Sense에서 다양한 z-test 함수의 사용 방법을 예시하는 목적으로는 충분합니다.

다음과 같이 하십시오.

1. 새 시트로 새 앱을 만듭니다.

   t-test 함수용으로 앱을 만든 경우, 이를 사용하여 함수에 사용할 새 시트를 만들 수 있습니다.

2. 데이터 로드 편집기에서 다음을 입력합니다.

```
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 '|');
```

이 로드 스크립트에서는 **crosstable**에 3개의 인수가 필요하기 때문에 **recno()**가 포함됩니다. 따라서 **recno()**는 간편하게 추가 인수를 제공합니다(이 경우, 각 행의 ID). 생략하는 경우, **Comparison** 샘플 값이 로드되지 않습니다.

3. 아이콘 을 클릭하여 데이터를 로드합니다.

z-test 테이블 만들기

다음과 같이 하십시오.

1. 데이터 로드 편집기에서 을 클릭하여 앱 보기로 이동한 다음 위에서 만든 시트를 클릭합니다. 시트 보기가 열립니다.
2. **시트 편집**을 클릭하여 시트를 편집합니다.
3. **차트**에서 테이블을 추가하고 **필드**에서 Type을 차원으로 추가합니다.
4. 테이블에 다음 표현식을 측정값으로 추가합니다.

표현식 예

레이블	표현식
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)

*의미 있는 값을 보기 위해 측정값의 숫자 형식 지정을 조정할 수 있습니다. 대부분의 측정값에서 숫자 서식을 **Auto** 대신 **숫자>단순**으로 설정하면 테이블을 읽기가 쉬워집니다. 그러나 예를 들어 ZTest Sig의 경우 숫자 형식을 사용합니다. **사용자 지정** 숫자 서식을 사용한 다음 서식 패턴을 **#.######**으로 조정합니다.*

결과:

샘플 데이터에 해당하는 z-test 함수의 결과 테이블에 다음과 같은 값이 포함됩니다.

z-test 결과 테이블

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

z-testw 테이블 만들기

z-testw 함수는 입력 데이터 수열이 가중치가 적용된 2열 형식으로 발생할 때 사용하기 위한 것입니다. 표현식에 인수 weight에 대한 값이 필요합니다.

여기의 예에서는 전체적으로 값 2를 사용하지만, 각 관찰에 대한 weight 값을 정의하는 표현식을 사용할 수도 있습니다.

다음과 같이 하십시오.

1. 데이터 로드 편집기에서 ⓔ을 클릭하여 앱 보기로 이동한 다음 위에서 만든 시트를 클릭합니다. 시트 보기가 열립니다.
2. **시트 편집**을 클릭하여 시트를 편집합니다.
3. **차트**에서 테이블을 추가하고 **필드**에서 Type을 차원으로 추가합니다.
4. 테이블에 다음 표현식을 측정값으로 추가합니다.

표현식 예

레이블	표현식
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)

z-test 함수 예와 동일한 숫자 형식 지정을 사용합니다.

결과:

z-testw 함수의 결과 테이블에는 다음 값이 포함됩니다.

z-testw 결과 테이블

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

문자열 집계 함수

이 섹션에서는 문자열 관련 집계 함수에 대해 설명합니다.

각 함수는 개요가 끝난 후에 더 자세히 설명합니다. 구문에서 함수 이름을 클릭하여 해당 함수에 대한 상세 설명에 즉시 액세스할 수도 있습니다.

데이터 로드 스크립트의 문자열 집계 함수

Concat

Concat()은 문자열 값을 결합하는 데 사용됩니다. 이 스크립트 함수는 **group by** 절로 정의된 여러 레코드에서 반복되는 표현식의 모든 값의 집계된 문자열 연결을 반환합니다.

```
Concat ([ distinct ] expression [, delimiter [, sort-weight]])
```

FirstValue

FirstValue()는 **group by** 절로 정렬되고 표현식으로 정의된 레코드에서 첫 번째로 로드된 값을 반환합니다.

> *이 함수는 스크립트 함수로만 사용할 수 있습니다.*

```
FirstValue  (expression)
```

LastValue

LastValue()는 **group by** 절로 정렬되고 표현식으로 정의된 레코드에서 마지막으로 로드된 값을 반환합니다.

> *이 함수는 스크립트 함수로만 사용할 수 있습니다.*

```
LastValue  (expression)
```

MaxString

MaxString()은 표현식 또는 필드에서 문자열 값을 찾아서 **group by** 절로 정의된 여러 레코드에 대해 알파벳 순으로 정렬된 마지막 텍스트 값을 반환합니다.

```
MaxString (expression )
```

MinString

MinString()은 표현식 또는 필드에서 문자열 값을 찾아서 **group by** 절로 정의된 여러 레코드에 대해 알파벳 순으로 정렬된 첫 번째 텍스트 값을 반환합니다.

```
MinString (expression )
```

차트 내의 문자열 집계 함수

다음과 같은 차트 함수를 차트에서 문자열을 집계하는 데 사용할 수 있습니다.

Concat

Concat()은 문자열 값을 결합하는 데 사용됩니다. 이 함수는 각 차원에서 평가된 표현식의 모든 값에 대해 집계된 문자열 연결을 반환합니다.

```
Concat - 차트 함수({[SetExpression] [DISTINCT] [TOTAL [<fld{, fld}>]] string[,
delimiter[, sort_weight]])
```

MaxString

MaxString()은 표현식 또는 필드에서 문자열 값을 찾아서 알파벳 정렬 순서의 마지막 텍스트 값을 반환합니다.

```
MaxString - 차트 함수({[SetExpression] [TOTAL [<fld{, fld}>]]} expr)
```

MinString

MinString()은 표현식 또는 필드에서 문자열 값을 찾아서 알파벳 정렬 순서의 첫 번째 텍스트 값을 반환합니다.

```
MinString - 차트 함수({[SetExpression] [TOTAL [<fld {, fld}>]]} expr)
```

Concat

Concat()은 문자열 값을 결합하는 데 사용됩니다. 이 스크립트 함수는 **group by** 절로 정의된 여러 레코드에서 반복되는 표현식의 모든 값의 집계된 문자열 연결을 반환합니다.

구문:

```
Concat ([ distinct ] string [, delimiter [, sort-weight]])
```

반환 데이터 유형: 문자열

인수:

처리할 문자열이 포함된 표현식 또는 필드입니다.

인수

인수	설명
string	처리할 문자열이 포함된 표현식 또는 필드입니다.
delimiter	각 값은 delimiter에 있는 문자열로 구분할 수 있습니다.
sort-weight	연결 순서는 연결에서 처음 나타나는 가장 낮은 값에 해당하는 문자열(있는 경우)이 포함된 **sort-weight** 차원의 값에 따라 결정할 수 있습니다..
distinct	표현식 앞에 **distinct**라는 단어가 있을 경우 모든 중복 항목이 무시됩니다.

예 및 결과:

예제 스크립트를 앱에 추가하고 실행합니다. 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.

예 및 결과

예	결과	시트에 추가된 후 결과
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
이전 예에서처럼 **TeamData** 테이블이 로드된 것으로 가정합니다. LOAD SalesGroup,Concat(distinct Team,'-') as TeamConcat2 Resident TeamData Group By SalesGroup;	SalesGroup East West	TeamConcat2 Alpha-Beta-Delta-Gamma Epsilon-Eta-Theta-Zeta
이전 예에서처럼 **TeamData** 테이블이 로드된 것으로 가정합니다. **sort-weight**에 대한 인수가 추가되었기 때문에 결과는 차원 Amount의 값에 따라 정렬됩니다. 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 - 차트 함수

Concat()은 문자열 값을 결합하는 데 사용됩니다. 이 함수는 각 차원에서 평가된 표현식의 모든 값에 대해 집계된 문자열 연결을 반환합니다.

구문:

```
Concat({[SetExpression] [DISTINCT] [TOTAL [<fld{, fld}>]]} string[, delimiter [, sort_weight]])
```

반환 데이터 유형: 문자열

인수:

인수

인수	설명
string	처리할 문자열이 포함된 표현식 또는 필드입니다.
delimiter	각 값은 delimiter에 있는 문자열로 구분할 수 있습니다.
sort-weight	연결 순서는 연결에서 처음 나타나는 가장 낮은 값에 해당하는 문자열(있는 경우)이 포함된 **sort-weight** 차원의 값에 따라 결정할 수 있습니다..
SetExpression	기본적으로 집계 함수는 선택에 의해 정의된 사용 가능한 레코드의 집합을 집계합니다. 집합 분석 표현식으로 대체 레코드 집합을 정의할 수 있습니다.
DISTINCT	함수 인수 앞에 **DISTINCT**라는 단어가 있을 경우 해당 함수 인수의 평가 결과로 생성된 중복이 무시됩니다.
TOTAL	**TOTAL**이 함수 인수 앞에 오는 경우, 현재 선택을 고려하되 현재 차원 값에 관련되지 않은 가능한 모든 값에 대한 계산이 실행됩니다. 즉, 차트 차원은 무시됩니다. **TOTAL [<fld {.fld}>]**(여기서 **TOTAL** 한정자 뒤에는 하나 이상의 필드 이름 목록이 차트 차원 변수의 하위 집합으로 옴)을 사용하여 가능한 전체 값의 하위 집합을 만듭니다.

예 및 결과:

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

함수 예

예	결과
Concat(Team)	이 테이블은 차원 SalesGroup 및 Amount, 측정값 Concat(Team)의 변형으로 구성됩니다. 합계 결과를 무시하면, SalesGroup의 두 값에 걸쳐 Team 값 여덟 개가 분포되어 있고, 테이블 내에 있는 둘 이상의 Team 문자열 값에 컨케트네이트되는 계수 Concat (Team)의 유일한 결과는 차원 Amount 20000이 포함된 행이며, 이는 결과 BetaGammaGamma를 제공합니다. 이는 입력 데이터에 Amount 20000에 해당하는 값이 세 개이기 때문입니다. SalesGroup 및 Amount의 각 조합에 단 하나의 Team 값만이 존재하므로 차원 전체에 계수를 확장할 때 다른 결과는 컨케트네이트되지 않는 채로 남습니다.
Concat (DISTINCT Team,', ')	Beta, Gamma이며, DISTINCT 한정자는 중복 Gamma 결과가 무시됨을 의미하기 때문입니다. 또한 구분 기호 인수는 쉼표에 이은 공백으로 정의됩니다.
Concat (TOTAL <SalesGroup> Team)	TOTAL 한정자를 사용하면 Team의 모든 값에 대한 문자열 값이 컨케트네이트됩니다. 필드 선택 <SalesGroup>이 지정된 경우, 이는 결과를 차원 SalesGroup의 두 값으로 나누게 됩니다. SalesGroupEast의 경우 결과가 AlphaBetaDeltaGammaGamma입니다. SalesGroupWest의 경우 결과가 EpsilonEtaThetaZeta입니다.
Concat (TOTAL <SalesGroup> Team,';', Amount)	**sort-weight**: Amount에 대한 인수를 추가하면 결과는 차원 Amount의 값에 따라 정렬됩니다. 결과는 DeltaBetaGammaGammaAlpha 및 EtaEpsilonZEtaTheta가 됩니다.

데이터 사용 예:

```
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()는 **group by** 절로 정렬되고 표현식으로 정의된 레코드에서 첫 번째로 로드된 값을 반환합니다.

이 함수는 스크립트 함수로만 사용할 수 있습니다.

구문:

```
FirstValue ( expr)
```

반환 데이터 유형: dual

인수:

인수

인수	설명
expr	측정할 데이터가 포함된 표현식 또는 필드입니다.

제한 사항:

텍스트 값이 발견되지 않으면 NULL이 반환됩니다.

예 및 결과:

예제 스크립트를 앱에 추가하고 실행합니다. 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.

결과 데이터

예	결과	시트에서의 결과
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()는 **group by** 절로 정렬되고 표현식으로 정의된 레코드에서 마지막으로 로드된 값을 반환합니다.

이 함수는 스크립트 함수로만 사용할 수 있습니다.

구문:

```
LastValue ( expr )
```

반환 데이터 유형: dual

인수:

인수

인수	설명
expr	측정할 데이터가 포함된 표현식 또는 필드입니다.

제한 사항:

텍스트 값이 발견되지 않으면 NULL이 반환됩니다.

예 및 결과:

예제 스크립트를 앱에 추가하고 실행합니다. 그런 다음, 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.

아래의 결과 열과 동일한 결과를 얻으려면 속성 패널의 정렬 아래에서 자동을 사용자 지정으로 전환한 후 숫자순 및 사전순 정렬을 선택 취소합니다.

예	결과	사용자 지정 정렬을 사용한 결과
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()은 표현식 또는 필드에서 문자열 값을 찾아서 **group by** 절로 정의된 여러 레코드에 대해 알파벳 순으로 정렬된 마지막 텍스트 값을 반환합니다.

구문:

```
MaxString ( expr )
```

반환 데이터 유형: dual

인수:

인수	설명
expr	측정할 데이터가 포함된 표현식 또는 필드입니다.

제한 사항:

텍스트 값이 발견되지 않으면 NULL이 반환됩니다.

예 및 결과:

예제 스크립트를 앱에 추가하고 실행합니다. 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.

예	결과	

`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

이전 예에서처럼 **TeamData** 테이블이 로드되고 데이터 로드 스크립트에는 SET 문이 있다고 가정합니다. `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 - 차트 함수

MaxString()은 표현식 또는 필드에서 문자열 값을 찾아서 알파벳 정렬 순서의 마지막 텍스트 값을 반환합니다.

구문:

```
MaxString({[SetExpression] [TOTAL [<fld{, fld}>]]} expr)
```

반환 데이터 유형: dual

인수:

인수

인수	설명
expr	측정할 데이터가 포함된 표현식 또는 필드입니다.
SetExpression	기본적으로 집계 함수는 선택에 의해 정의된 사용 가능한 레코드의 집합을 집계합니다. 집합 분석 표현식으로 대체 레코드 집합을 정의할 수 있습니다.
TOTAL	**TOTAL**이 함수 인수 앞에 오는 경우, 현재 선택을 고려하되 현재 차원 값에 관련되지 않은 가능한 모든 값에 대한 계산이 실행됩니다. 즉, 차트 차원은 무시됩니다. **TOTAL [<fld {.fld}>]**(여기서 **TOTAL** 한정자 뒤에는 하나 이상의 필드 이름 목록이 차트 차원 변수의 하위 집합으로 옴)을 사용하여 가능한 전체 값의 하위 집합을 만듭니다.

제한 사항:

표현식에 문자열 표현이 포함된 값이 없다면 NULL이 반환됩니다.

예 및 결과:

결과 테이블

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

함수 예

예	결과
`MaxString (Team)`	차원 Amount에는 Gamma 두 개(서로 다른 날짜), Beta 한 개 등 세 개의 20000 값이 있습니다. 따라서 측정값 MaxString (Team)의 결과는 정렬된 문자열 중에서 가장 높은 값인 Gamma가 됩니다.
`MaxString (Date)`	2013/11/01은 차원 Amount와 연결된 셋 중에서 가장 큰 Date 값입니다. 이는 사용자의 스크립트에 SET 문 `SET DateFormat='YYYY-MM-DD';`가 있음을 가정한 것입니다.

데이터 사용 예:

```
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()은 표현식 또는 필드에서 문자열 값을 찾아서 **group by** 절로 정의된 여러 레코드에 대해 알파벳 순으로 정렬된 첫 번째 텍스트 값을 반환합니다.

구문:

```
MinString ( expr )
```

반환 데이터 유형: dual

인수:

인수

인수	설명
expr	측정할 데이터가 포함된 표현식 또는 필드입니다.

제한 사항:

텍스트 값이 발견되지 않으면 NULL이 반환됩니다.

예 및 결과:

예제 스크립트를 앱에 추가하고 실행합니다. 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.

결과 데이터

예	결과	
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
이전 예에서처럼 **TeamData** 테이블이 로드되고 데이터 로드 스크립트에는 SET 문이 있다고 가정합니다. 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 - 차트 함수

MinString()은 표현식 또는 필드에서 문자열 값을 찾아서 알파벳 정렬 순서의 첫 번째 텍스트 값을 반환합니다.

구문:

```
MinString({[SetExpression] [TOTAL [<fld {, fld}>]]} expr)
```

반환 데이터 유형: dual

인수:

인수

인수	설명
expr	측정할 데이터가 포함된 표현식 또는 필드입니다.
SetExpression	기본적으로 집계 함수는 선택에 의해 정의된 사용 가능한 레코드의 집합을 집계합니다. 집합 분석 표현식으로 대체 레코드 집합을 정의할 수 있습니다.

인수	설명
TOTAL	**TOTAL**이 함수 인수 앞에 오는 경우, 현재 선택을 고려하되 현재 차원 값에 관련되지 않은 가능한 모든 값에 대한 계산이 실행됩니다. 즉, 차트 차원은 무시됩니다. **TOTAL [<fld {.fld}>]**(여기서 **TOTAL** 한정자 뒤에는 하나 이상의 필드 이름 목록이 차트 차원 변수의 하위 집합으로 옴)을 사용하여 가능한 전체 값의 하위 집합을 만듭니다.

예 및 결과:

샘플 데이터

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

함수 예

예	결과
`MinString (Team)`	차원 Amount에는 Gamma 두 개(서로 다른 날짜), Beta 한 개 등 세 개의 20000 값이 있습니다. 따라서 측정값 MinString (Team)의 결과는 정렬된 문자열 중에서 첫 번째 값인 Beta가 됩니다.
`MinString (Date)`	2013/11/01은 차원 Amount와 연결된 셋 중에서 가장 빠른 Date 값입니다. 이는 사용자의 스크립트에 SET 문 `SET DateFormat='YYYY-MM-DD';`가 있음을 가정한 것입니다.

데이터 사용 예:

```
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 '|');
```

가상 차원 함수

가상 차원은 가상 차원 함수에서 생성된 값으로 앱에서 만들어지며, 데이터 모델에 포함된 필드에서 직접 만들어지지 않습니다. 가상 차원 함수에서 생성된 값을 차트에 계산 차원으로 사용하는 경우, 가상 차원이 만들어집니다.. 가상 차원을 사용하면 데이터에서 발생되는 값으로 차원(동적 차원)이 포함된 차트를 만들 수 있습니다.

가상 차원은 선택 내용의 영향을 받지 않습니다.

다음 가상 차원 함수를 차트에서 사용할 수 있습니다.

ValueList

ValueList()는 계산 차원에 사용할 경우 가상 차원을 형성하는 나열된 값의 집합을 반환합니다.

```
ValueList - 차트 함수 (v1 {, Expression})
```

ValueLoop

ValueLoop()는 계산 차원에 사용할 경우 가상 차원을 형성하는 반복 값의 집합을 반환합니다.

```
ValueLoop - 차트 함수(from [, to [, step ]])
```

ValueList - 차트 함수

ValueList()는 계산 차원에 사용할 경우 가상 차원을 형성하는 나열된 값의 집합을 반환합니다.

***ValueList** 함수를 사용하여 만든 가상 차원이 있는 차트에서는 차트 표현식에 **ValueList** 함수에 동일한 매개 변수를 다시 지정하여 특정 표현식 셀에 해당하는 차원 값을 참조할 수 있습니다. 이 함수는 물론 레이아웃의 어느 곳에서나 사용할 수 있지만 가상 차원에 사용되는 경우를 제외하면 집계 함수 내에서만 의미가 있습니다.*

가상 차원은 선택 내용의 영향을 받지 않습니다.

구문:

```
ValueList(v1 {,...})
```

반환 데이터 유형: dual

인수:

인수

인수	설명
v1	정적 값(일반적으로 문자열이지만 숫자가 될 수도 있음).
{,...}	정적 값의 선택적 목록.

예 및 결과:

함수 예

예	결과
`ValueList ('Number of Orders', 'Average Order Size', 'Total Amount')`	예를 들어 테이블에서 차원을 만드는 데 사용할 경우, 세 문자열 값이 테이블의 행 레이블이 됩니다. 그러면 이를 표현식에서 참조할 수 있습니다.
`=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) ))`	이 표현식은 생성된 차원에서 값을 가져와 중첩된 IF 문에 다음 세 집계 함수에 대한 입력으로 이를 참조합니다. **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

예에서 사용된 데이터:

```
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 - 차트 함수

ValueLoop()는 계산 차원에 사용할 경우 가상 차원을 형성하는 반복 값의 집합을 반환합니다.
생성된 값은 **from** 값에서 시작하여 **to** 값으로 끝나며, 단계별로 증가되는 중간 값이 포함됩니다.

ValueLoop 함수를 사용하여 만든 가상 차원이 있는 차트에서는 차트 표현식에 ValueLoop 함수에 동일한 매개 변수를 다시 지정하여 특정 표현식 셀에 해당하는 차원 값을 참조할 수 있습니다. 이 함수는 물론 레이아웃의 어느 곳에서나 사용할 수 있지만 가상 차원에 사용되는 경우를 제외하면 집계 함수 내에서만 의미가 있습니다.

가상 차원은 선택 내용의 영향을 받지 않습니다.

구문:

```
ValueLoop(from [, to [, step ]])
```

반환 데이터 유형: dual

인수:

인수

인수	설명
from	생성될 값 집합의 시작 값입니다.
to	생성될 값 집합의 종료 값입니다.
step	값 사이의 증분 크기입니다.

예 및 결과:

함수 예

예	결과
`ValueLoop (1, 10)`	이는 테이블에 예를 들어 번호로 된 레이블을 지정하는 등에 사용할 수 있는 차원을 생성합니다. 여기의 예에서는 값에 1에서 10까지 번호가 지정되었습니다. 그러면 이 값을 표현식에서 참조할 수 있습니다.
`ValueLoop (2, 10,2)`	이 예에서는 인수 step의 값이 2이므로 값에 2, 4, 6, 8, 10의 번호가 매겨집니다.

중첩 집계

다른 집계의 결과에 집계를 적용해야 하는 상황이 있을 수 있습니다. 이를 중첩 집계라고 부릅니다.

대부분의 차트 표현식에서는 집계를 중첩할 수 없습니다. 그러나 내부 집계 함수에서 **TOTAL** 한정자를 사용하는 경우 집계를 중첩할 수 있습니다.

100개를 초과하는 중첩 수준은 허용되지 않습니다.

TOTAL 한정자를 사용한 중첩 집계

예를 들어, **Sales** 필드의 합을 계산하되 **OrderDate**가 지난 해와 일치하는 트랜잭션만 포함되도록 만들려는 경우가 있습니다. 지난 해는 집계 함수 **Max(TOTAL** `Year(OrderDate)`**)**를 통해 구할 수 있습니다.

다음 집계는 원하는 결과를 반환합니다.

```
Sum(If(Year(OrderDate)=Max(TOTAL Year(OrderDate)), Sales))
```

Qlik Sense는 이 중첩 유형의 **TOTAL** 한정자를 포함해야 합니다. 원하는 비교를 위해 필요합니다. 이러한 유형의 중첩 요구 사항은 상당히 일반적이며 적절한 조치입니다.

관련 항목:

Aggr - 차트 함수 (page 524)

8.3 Aggr - 차트 함수

Aggr()은 지정된 하나 또는 복수의 차원에서 계산된 표현식에 대한 값의 배열을 반환합니다. 판매량, 고객 및 지역별 최대값을 예로 들 수 있습니다.

Aggr 함수는 중첩 집계에 사용되며 첫 번째 매개 변수(내부 집계)는 차원 값마다 한 번 계산됩니다. 차원은 두 번째 매개 변수(및 후속 매개 변수)에서 지정됩니다.

또한 **Aggr** 함수는 중첩된 집계에 대한 입력으로 **Aggr** 함수의 결과로 구성된 배열을 사용하는 외부 집계 함수 안에 포함되어야 합니다.

구문:

```
Aggr({SetExpression}[DISTINCT] [NODISTINCT ] expr, StructuredParameter{,
StructuredParameter})
```

반환 데이터 유형: dual

인수:

인수

인수	설명
expr	집계 함수로 구성된 표현식. 기본적으로 집계 함수는 선택에 의해 정의된 사용 가능한 레코드의 집합을 집계합니다.

인수	설명
StructuredParameter	StructuredParameter는 차원과 형식의 정렬 기준(옵션)으로 이루어집니다. `(Dimension(Sort-type, Ordering))` 차원은 단일 필드이며 표현식이 될 수 없습니다. 차원은 Aggr 표현식으로 계산되는 값의 배열을 결정하는 데 사용됩니다. 정렬 기준이 포함된 경우, 차원에 대해 계산되고 Aggr 함수에 의해 생성된 값의 배열이 정렬됩니다. Aggr 함수가 포함된 표현식의 결과에 정렬 순서가 영향을 미치는 경우에 중요합니다. 정렬 기준 사용 방법에 대한 자세한 내용은 구조화된 매개 변수로 차원에 정렬 기준 추가를 참조하십시오.
SetExpression	기본적으로 집계 함수는 선택에 의해 정의된 사용 가능한 레코드의 집합을 집계합니다. 집합 분석 표현식으로 대체 레코드 집합을 정의할 수 있습니다.
DISTINCT	표현식 인수에 **distinct** 한정자가 선행하는 경우 또는 아무 한정자도 사용하지 않는 경우 각 차원 값의 조합에서 하나의 반환 값만 생성됩니다. 이는 집계가 구성되는 일반적인 방식이며, 각 차원 값의 고유 조합이 차트에서 한 줄을 이룹니다.
NODISTINCT	표현식 인수에 **nodistinct** 한정자가 선행하는 경우 원본 데이터 구조에 따라 각 차원 값의 조합에서 2개 이상의 반환 값이 생성될 수 있습니다. 차원이 하나뿐이라면 소스 데이터에 행이 있으므로 **aggr** 함수는 동일한 요소 수의 배열을 반환합니다.

Sum, **Min**, **Avg**와 같은 기본 집계 함수는 단일 숫자 값을 반환하지만, Aggr() 함수는 다른 집계를 만들 수 있는 임시 계획 결과 집합(가상 테이블)을 만드는 것과 비교할 수 있습니다. 예를 들어, **Aggr()** 문에서 판매량을 고객과 합산하여 평균 판매량 값을 계산하고 합산된 결과의 평균을 계산하는 것입니다. **Avg(TOTAL Aggr (Sum(Sales),Customer))**.

Aggr() 함수를 계산된 차원에서 사용하여 다양한 수준의 중첩 차트 집계를 생성할 수 있습니다.

제한 사항:

Aggr() 함수의 각 차원은 단일 필드여야 하며 표현식(계산된 차원)이 될 수 없습니다.

구조화된 매개 변수로 차원에 정렬 기준 추가

Aggr 함수 구문에서 StructuredParameter 인수의 기본 형태는 단일 차원입니다. 표현식: Aggr(Sum(Sales, Month))는 매월 판매액의 총액을 찾습니다. 그러나 다른 집계 함수에 포함된 경우 정렬 기준을 사용하지 않으면 예상치 않은 결과가 나올 수 있습니다. 일부 차원은 숫자순이나 사전순 등으로 정렬될 수 있기 때문입니다.

Aggr 함수의 StructuredParameter 인수에서 표현식의 차원에 대한 정렬 기준을 지정할 수 있습니다. 이 방법으로 Aggr 함수에 의해 생성된 가상 테이블에 정렬 기준을 적용할 수 있습니다.

인수 StructuredParameter에는 다음과 같은 구문이 있습니다.

```
(FieldName, (Sort-type, Ordering))
```

구조화된 매개 변수는 중첩할 수 있습니다.

```
(FieldName, (FieldName2, (Sort-type, Ordering)))
```

Sort-type은 NUMERIC, TEXT, FREQUENCY 또는 LOAD_ORDER일 수 있습니다.

Ordering 유형은 다음과 같이 각 Sort-type과 연결됩니다.

허용되는 Ordering 유형

Sort-type	**허용되는 Ordering 유형**
NUMERIC	ASCENDING, DESCENDING 또는 REVERSE
TEXT	ASCENDING, A2Z, DESCENDING, REVERSE 또는 Z2A
FREQUENCY	DESCENDING, REVERSE 또는 ASCENDING
LOAD_ORDER	ASCENDING, ORIGINAL, DESCENDING 또는 REVERSE

Ordering 유형 REVERSE와 DESCENDING은 동일합니다.

Sort-type TEXT의 경우 Ordering 유형 ASCENDING과 A2Z가 동일하고, DESCENDING, REVERSE 및 Z2A가 동일합니다.

Sort-type LOAD_ORDER의 경우 Ordering 유형 ASCENDING과 ORIGINAL이 동일합니다.

예: Aggr을 사용한 차트 표현식

예 - 차트 표현식

차트 표현식 예 1

로드 스크립트

데이터 로드 편집기에서 다음 데이터를 인라인 로드로 로드하여 아래 차트 표현식 예를 만듭니다.

```
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 '|');
```

차트 표현식

Qlik Sense 시트에 KPI 시각화를 만듭니다. 다음 표현식을 측정값으로 KPI에 추가합니다.

```
Avg(Aggr(Sum(UnitSales*UnitPrice), Customer))
```

결과

376.7

설명

표현식 `Aggr(Sum(UnitSales*UnitPrice), Customer)`은 판매액의 총 값을 **Customer**로 찾은 다음 3개의 **Customer** 값에 대해 295, 715 및 120의 값 배열을 반환합니다.

실제로는 해당 값을 포함하는 명시적인 테이블이나 열을 만들지 않고 임시 값 목록을 작성했습니다.

이 값은 판매량의 평균 값 376.7을 찾기 위해 **Avg()** 함수에 입력으로 사용됩니다.

차트 표현식 예 2

로드 스크립트

데이터 로드 편집기에서 다음 데이터를 인라인 로드로 로드하여 아래 차트 표현식 예를 만듭니다.

```
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 '|');
```

차트 표현식

Customer, **Product**, **UnitPrice**, **UnitSales**가 차원으로 포함된 Qlik Sense 시트에 테이블 시각화를 만듭니다. 테이블에 다음 표현식을 측정값으로 추가합니다.

```
Aggr(NODISTINCT Max(UnitPrice), Customer, Product)
```

결과

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

Customer	Product	UnitPrice	UnitSales	Aggr(NODISTINCT Max(UnitPrice), Customer, Product)
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

설명

값의 배열은 16, 16, 15, 15, 12, 12, 22, 22, 25, 15, 15 및 19입니다. **nodistinct** 한정자는 배열이 소스 데이터의 각 행에 대해 하나의 요소를 포함하고 있음을 의미합니다. 각각은 각 **Customer** 및 **Product**에 대한 최대 **UnitPrice**입니다.

차트 표현식 예 3

로드 스크립트

데이터 로드 편집기에서 다음 데이터를 인라인 로드로 로드하여 아래 차트 표현식 예를 만듭니다.

```
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;
```

차트 표현식

Year와 **Month**가 차원으로 포함된 Qlik Sense 시트에 테이블 시각화를 만듭니다. 테이블에 다음 표현식을 측정값으로 추가합니다.

- `Sum(Sales)`
- 테이블에서 Structured Aggr()로 `Sum(Aggr( Rangesum(Above(Sum(Sales),0,12)), (Year, (Numeric, Ascending)), (Month, (Numeric, Ascending)) ))` 레이블이 지정됩니다.

결과

Year	Month	Sum(Sales)	Structured Aggr()
2013	Jan	53495	53495
2013	Feb	48580	102075
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

설명

이 예에서는 각 연도에 대해 12개월 동안 집계된 값을 시간 오름차순으로 표시하므로 구조화된 매개 변수(숫자, 오름차순)는 **Aggr()** 표현식의 일부입니다. 구조화된 매개 변수로 두 가지 특정 차원이 필요합니다. **Year** 및 **Month**, 정렬된 (1) **Year**(숫자) 및 (2) **Month**(숫자)입니다. 이 두 차원은 테이블 또는 차트 시각화에서 사용해야 합니다. 이를 통해 **Aggr()** 함수의 차원 목록이 시각화에 사용된 개체의 차원과 일치할 수 있습니다.

이러한 측정값 간의 차이를 테이블 또는 별도의 꺾은선형 차트에서 비교할 수 있습니다.

- `Sum(Aggr( Rangesum(Above(Sum(Sales),0,12)), (Year), (Month) ))`
- `Sum(Aggr( Rangesum(Above(Sum(Sales),0,12)), (Year, (Numeric, Ascending)), (Month, (Numeric, Ascending)) ))`

후자의 표현식만이 집계 값을 원하는 대로 누적할 수 있음이 명확히 표시되어야 합니다.

관련 항목:

기본 집계 함수 (page 318)

8.4 색 함수

차트 개체와 함께 데이터 로드 스크립트의 색 속성을 설정하고 평가하는 것과 관련된 표현식에 사용할 수 있는 함수입니다.

Qlik Sense는 이전 버전과의 호환성을 위해 색 함수 ***Color()***, ***qliktechblue*** *및* ***qliktechgray****를 지원하지만 사용하지 않는 것이 좋습니다.*

ARGB

ARGB()는 표현식에서 빨강 성분 **r**, 녹색 성분 **g**, 파랑 성분 **b** 및 **alpha**의 알파 요소(불투명도)로 색이 정의된 차트 개체의 색 속성을 설정하거나 평가하는 데 사용됩니다.

```
ARGB (alpha, r, g, b)
```

HSL

HSL()은 표현식에서 0 ~ 1 사이의 **hue**, **saturation** 및 **luminosity** 값으로 색이 정의된 차트 개체의 색 속성을 설정하거나 평가하는 데 사용됩니다.

```
HSL  (hue, saturation, luminosity)
```

RGB

RGB()는 빨간색 구성 요소 r, 녹색 구성 요소 g 및 파란색 구성 요소 b의 세 가지 매개 변수로 정의된 색상의 색상 코드에 해당하는 정수를 반환합니다. 이러한 구성 요소는 0에서 255 사이의 정수 값을 가져야 합니다. 이 함수는 차트 개체의 색상 속성을 설정하거나 평가하기 위해 표현식에서 사용할 수 있습니다.

```
RGB  (r, g, b)
```

Colormix1

Colormix1()은 표현식에서 0과 1 사이의 값을 기준으로 2색 그라데이션에서 ARGB 색 표현을 반환하는 데 사용됩니다.

```
Colormix1 (Value , ColorZero , ColorOne)
```

Value는 0과 1 사이의 실수입니다.

- Value = 0이면 ColorZero 가 반환됩니다.
- Value = 1이면 ColorOne 이 반환됩니다.

- 0 < Value< 1이면 적절한 중간 음영이 반환됩니다.

ColorZero는 간격의 저점과 연결될 색의 유효한 RGB 색 표현입니다.

ColorOne는 간격의 고점과 연결될 색의 유효한 RGB 색 표현입니다.

```
Colormix1(0.5, red(), blue())
```
반환 값:

```
ARGB(255,64,0,64) (purple)
```

Colormix2

Colormix2()는 표현식에서 -1과 1 사이의 값을 기준으로 2색 그라데이션에서 ARGB 색 표현을 반환하는 데 사용되며 중앙(0) 위치에 대한 중간 값을 지정할 수 있습니다.

```
Colormix2 (Value ,ColorMinusOne , ColorOne[ , ColorZero])
```
Value는 -1과 1 사이의 실수입니다.

- Value = -1이면 첫 번째 색이 반환됩니다.
- Value = 1이면 두 번째 색이 반환됩니다.
- -1 < Value< 1이면 적절한 색 조합이 반환됩니다.

ColorMinusOne는 간격의 저점과 연결될 색의 유효한 RGB 색 표현입니다.

ColorOne는 간격의 고점과 연결될 색의 유효한 RGB 색 표현입니다.

ColorZero는 간격의 중앙과 연결될 색의 유효한 RGB 색 표현이며 옵션입니다.

SysColor

SysColor()는 Windows 시스템 색 nr에 해당하는 ARGB 색 표현을 반환하며, 여기서 nr은 Windows API 함수 **GetSysColor(nr)**에 대한 파라메타에 해당합니다.

```
SysColor (nr)
```

ColorMapHue

ColorMapHue()는 HSV 컬러 모델의 색상 구성 요소가 다양한 컬러맵의 한 색의 ARGB 값을 반환합니다. 컬러맵은 빨강으로 시작하여, 노랑, 녹색, 녹청색, 파랑, 자홍색을 거쳐 다시 빨강으로 돌아옵니다. x는 0에서 1 사이의 값이어야 합니다.

```
ColorMapHue (x)
```

ColorMapJet

ColorMapJet()은 파랑으로 시작하여, 녹청색, 노랑, 주황색을 거쳐 다시 빨강으로 돌아오는 컬러맵의 한 색의 ARGB 값을 반환합니다. x는 0에서 1 사이의 값이어야 합니다.

```
ColorMapJet (x)
```

사전 정의 색 함수

다음은 표현식에서 사전 정의 색에 대해 사용할 수 있는 함수입니다. 각 함수는 RGB 색 표현을 반환합니다.

또는 알파 요소에 해당하는 파라메타를 지정할 수 있으며, 이 경우 ARGB 색 표현이 반환됩니다. 알파 요소 0은 완전 투명에 해당하며 알파 요소 255는 완전 불투명에 해당합니다. alpha 값을 입력하지 않으면 255가 사용됩니다.

사전 정의 색 함수

색 함수	RGB 값
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)

예 및 결과:

예 및 결과

예	결과
`Blue()`	RGB(0,0,128)
`Blue(128)`	ARGB(128,0,0,128)

ARGB

ARGB()는 표현식에서 빨강 성분 **r**, 녹색 성분 **g**, 파랑 성분 **b** 및 **alpha**의 알파 요소(불투명도)로 색이 정의된 차트 개체의 색 속성을 설정하거나 평가하는 데 사용됩니다.

구문:

```
ARGB(alpha, r, g, b)
```

반환 데이터 유형: dual

인수:

인수

인수	설명
alpha	투명도 값은 0 - 255 범위입니다. 0은 완전 투명에 해당하며 255는 완전 불투명에 해당합니다.
r, g, b	빨강, 녹색 및 파랑 성분 값. 색 성분 0은 기여도가 없으며, 255부터 전체 기여도 중 하나입니다.

모든 인수는 0 ~ 255 범위의 정수로 해석되는 표현식이어야 합니다.

숫자 성분을 해석하고 16진수 표기법으로 서식을 지정하면 색 성분 값을 더 쉽게 파악할 수 있습니다. 예를 들어 연한 녹색의 숫자가 4 278 255 360인 경우, 이를 16진수 표기법으로 표현하면 FF00FF00입니다. 처음 두 위치 'FF'(255)는 **alpha** 채널을 나타냅니다. 다음 두 위치 '00'은 **빨강**의 양을 나타내고, 그 다음 두 위치 'FF'는 **초록**의 양을 나타내고, 마지막 두 위치 '00'은 **파랑**의 양을 나타냅니다.

RGB

RGB()는 빨간색 구성 요소 r, 녹색 구성 요소 g 및 파란색 구성 요소 b의 세 가지 매개 변수로 정의된 색상의 색상 코드에 해당하는 정수를 반환합니다. 이러한 구성 요소는 0에서 255 사이의 정수 값을 가져야 합니다. 이 함수는 차트 개체의 색상 속성을 설정하거나 평가하기 위해 표현식에서 사용할 수 있습니다.

구문:

```
RGB (r, g, b)
```

반환 데이터 유형: dual

인수:

인수

인수	설명
r, g, b	빨강, 녹색 및 파랑 성분 값. 색 성분 0은 기여도가 없으며, 255부터 전체 기여도 중 하나입니다.

모든 인수는 0 ~ 255 범위의 정수로 해석되는 표현식이어야 합니다.

숫자 성분을 해석하고 16진수 표기법으로 서식을 지정하면 색 성분 값을 더 쉽게 파악할 수 있습니다. 예를 들어 연한 녹색의 숫자가 4 278 255 360인 경우, 이를 16진수 표기법으로 표현하면 FF00FF00입니다. 처음 두 위치 'FF'(255)는 **alpha** 채널을 나타냅니다. **RGB** 및 **HSL** 함수에서는 항상 'FF'(불투명)입니다. 다음 두 위치 '00'은 **빨강**의 양을 나타내고, 그 다음 두 위치 'FF'는 **초록**의 양을 나타내고, 마지막 두 위치 '00'은 **파랑**의 양을 나타냅니다.

예: 차트 표현식
이 예에서는 차트에 사용자 지정 색을 적용합니다.

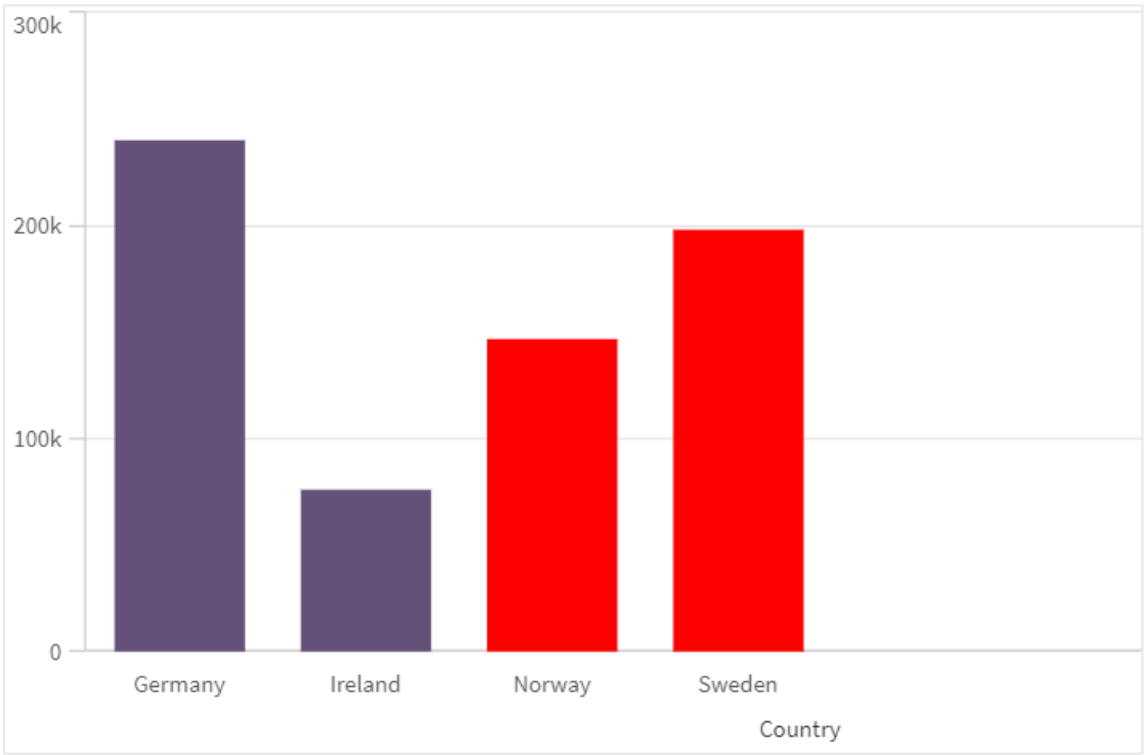
이 예에서 사용된 데이터:

```
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 ',');
```

색상 및 범례 속성 패널에 다음 표현식을 입력합니다.

```
If (Sum(Sales)>Sum(Budget),RGB(255,0,0),RGB(100,80,120))
```

결과:



예: 로드 스크립트
다음 예는 16진수 형식의 값에 해당하는 RGB 값을 표시합니다.

```
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];
```
결과:

텍스트	색
010203	RGB(1,2,3)
AABBCC	RGB(170,187,204)

HSL

HSL()은 표현식에서 0 ~ 1 사이의 **hue**, **saturation** 및 **luminosity** 값으로 색이 정의된 차트 개체의 색 속성을 설정하거나 평가하는 데 사용됩니다.

구문:

```
HSL (hue, saturation, luminosity)
```

반환 데이터 유형: dual

인수:

인수

인수	설명
hue, saturation, luminosity	hue, saturation, luminosity 성분 값은 0에서 1사이의 범위입니다.

모든 인수는 0 ~ 1 범위의 정수로 해석되는 표현식이어야 합니다.

숫자 성분을 해석하고 16진수 표기법으로 서식을 지정하면 색 성분의 RGB 값을 더 쉽게 파악할 수 있습니다. 예를 들어, 연한 녹색의 번호는 4 278 255 360이며, 이를 16진수 표기법으로 표현하면 FF00FF00 및 RGB (0,255,0)입니다. 이는 HSL (80/240, 240/240, 120/240) 즉, HSL 값 (0.33, 1, 0.5)와 같습니다.

8.5 조건부 함수

모든 조건부 함수가 조건을 평가한 다음, 조건 값에 따라 서로 다른 대답을 반환합니다. 함수는 데이터 로드 스크립트와 차트 표현식에서 사용할 수 있습니다.

조건부 함수 개요

각 함수는 개요가 끝난 후에 더 자세히 설명합니다. 구문에서 함수 이름을 클릭하여 해당 함수에 대한 상세 설명에 즉시 액세스할 수도 있습니다.

alt

alt 함수는 유효한 숫자 표현이 있는 첫 번째 파라메타를 반환합니다. 이와 일치하는 항목이 발견되지 않을 경우 마지막 파라메타를 반환합니다. 원하는 만큼의 파라메타를 사용할 수 있습니다.

```
alt (expr1[ , expr2 , expr3 , ...] , else)
```

class

class 함수는 첫 번째 파라메타를 클래스 간격에 할당합니다. 결과는 숫자 값으로 하위 경계, 텍스트 값으로 a<=x<b이 포함된 이중 값이며, 여기서 a와 b는 bin의 상한과 하한 값입니다.

```
class (expression, interval [ , label [ , offset ]])
```

coalesce

coalesce 함수는 유효한 non-NULL 표현이 있는 첫 번째 매개변수를 반환합니다. 원하는 만큼의 파라메타를 사용할 수 있습니다.

```
coalesce(expr1[ , expr2 , expr3 , ...])
```

if

if 함수는 함수에 제공된 조건이 True로 평가되는지, False로 평가되는지에 따른 값을 반환합니다.

```
if (condition , then , else)
```

match

match 함수는 첫 번째 파라메타를 이후의 모든 파라메타와 비교하여 일치하는 표현식의 수 위치를 반환합니다. 대/소문자가 구분됩니다.

```
match ( str, expr1 [ , expr2,...exprN ])
```

mixmatch

mixmatch 함수는 첫 번째 파라메타를 이후의 모든 파라메타와 비교하여 일치하는 표현식의 수 위치를 반환합니다. 대/소문자는 구분되지 않습니다.

```
mixmatch ( str, expr1 [ , expr2,...exprN ])
```

pick

pick 함수는 목록 내 *n*번째 표현식을 반환합니다.

```
pick (n, expr1[ , expr2,...exprN])
```

wildmatch

wildmatch 함수는 첫 번째 파라메타를 이후의 모든 파라메타와 비교하여 일치하는 표현식의 수를 반환합니다. 비교 문자열에서 와일드카드 문자(***** 및 **?**)의 사용을 허용합니다. *****는 문자의 순서와 일치합니다. **?**는 단일 문자와 일치합니다. 대/소문자는 구분되지 않습니다.

```
wildmatch ( str, expr1 [ , expr2,...exprN ])
```

alt

alt 함수는 유효한 숫자 표현이 있는 첫 번째 파라메타를 반환합니다. 이와 일치하는 항목이 발견되지 않을 경우 마지막 파라메타를 반환합니다. 원하는 만큼의 파라메타를 사용할 수 있습니다.

구문:

```
alt(expr1[ , expr2 , expr3 , ...] , else)
```

인수:

인수

인수	설명
expr1	유효한 숫자 표현인지 확인할 첫 번째 표현식입니다.
expr2	유효한 숫자 표현인지 확인할 두 번째 표현식입니다.
expr3	유효한 숫자 표현인지 확인할 세 번째 표현식입니다.
else	유효한 숫자 표현을 가진 이전 매개 변수가 없는 경우 반환하는 값입니다.

alt 함수는 숫자 또는 날짜 해석 함수와 함께 사용되는 경우가 많습니다. 이 방법으로 Qlik Sense에서 우선 순위에 따라 서로 다른 날짜 서식을 테스트할 수 있습니다. 또한 이 함수는 숫자 표현식에서 NULL 값을 처리하는 데도 사용할 수 있습니다.

예

예	결과

`alt( date#( dat , 'YYYY/MM/DD' ),`
 	
`date#( dat , 'MM/DD/YYYY' ),`
 	
`date#( dat , 'MM/DD/YY' ),`
 	
`'No valid date' )`
	이 표현식은 지정된 세 가지 날짜 형식에 해당하는 날짜가 날짜 필드에 포함되어 있는지 테스트합니다. 포함되어 있다면 원래 문자열과 날짜의 유효한 숫자 표현이 포함된 이중 값이 반환됩니다. 일치 항목이 발견되지 않을 경우 'No valid date'가 반환됩니다(유효한 숫자 표현 없이).

`alt(Sales,0) + alt(Margin,0)`	이 표현식은 Sales 및 Margin 필드를 추가하며, 누락된 모든 값(NULL)을 0으로 바꿉니다.

class

class 함수는 첫 번째 파라메타를 클래스 간격에 할당합니다. 결과는 숫자 값으로 하위 경계, 텍스트 값으로 a<=x<b이 포함된 이중 값이며, 여기서 a와 b는 bin의 상한과 하한 값입니다.

구문:

```
class(expression, interval [ , label [ , offset ]])
```

인수:

인수

인수	설명
interval	bin 너비를 지정하는 숫자입니다.
label	결과 텍스트에서 'x'를 대체할 수 있는 임의의 문자열입니다.
offset	기본 분류 시작점에서의 오프셋으로 사용할 수 있는 숫자입니다. 기본 시작점은 일반적으로 0입니다.

예

예	결과
`class( var,10 ), var = 23`	반환 값: '20<=x<30'
`class( var,5,'value' ), var = 23`	반환 값: '20<= value <25'
`class( var,10,'x',5 ), var = 23`	반환 값: '15<=x<25'

예 - class를 사용한 로드 스크립트

예: 로드 스크립트

로드 스크립트

이 예에서는 개인의 이름과 나이가 포함된 테이블을 로드하며 10세 간격으로 연령대에 따라 각 개인을 분류하는 필드를 추가하려고 합니다. 원본 소스 테이블은 다음과 같습니다.

결과

Name	Age
John	25
Karen	42
Yoshi	53

연령대 분류 필드를 추가하려면 **class** 함수를 사용하여 선행 load 문을 추가할 수 있습니다.

데이터 로드 편집기에서 새 탭을 만든 후 다음 데이터를 인라인 로드로 로드합니다. 결과를 보려면 Qlik Sense에서 아래 테이블을 만듭니다.

```
LOAD *,
class(Age, 10, 'age') As Agegroup;

LOAD * INLINE
```

```
[ Age, Name
25, John
42, Karen
53, Yoshi];
```

결과

결과

Name	Age	Agegroup
John	25	20 <= age < 30
Karen	42	40 <= age < 50
Yoshi	53	50 <= age < 60

coalesce

coalesce 함수는 유효한 non-NULL 표현이 있는 첫 번째 매개변수를 반환합니다. 원하는 만큼의 파라메타를 사용할 수 있습니다.

구문:

```
coalesce(expr1[ , expr2 , expr3 , ...])
```

인수:

인수

인수	설명
expr1	NULL이 아닌 유효한 표현인지 확인할 첫 번째 표현식입니다.
expr2	NULL이 아닌 유효한 표현인지 확인할 두 번째 표현식입니다.
expr3	NULL이 아닌 유효한 표현인지 확인할 세 번째 표현식입니다.

예

예	결과
	이 표현식은 필드의 모든 NULL 값을 'N/A'로 변경합니다.
`Coalesce(ProductDescription, ProductName, ProductCode, 'no description available')`	이 표현식은 일부 필드에 제품 값이 없는 경우 세 가지 다른 제품 설명 필드 중에서 선택합니다. Null 값이 아닌 첫 번째 필드가 지정된 순서대로 반환됩니다. 값이 있는 필드가 없는 경우 결과는 '사용 가능한 설명 없음'입니다.
`Coalesce(TextBetween(FileName, '"', '"'), FileName)`	이 표현식은 *FileName* 필드에서 잠재적인 인용 부호를 제거합니다. 주어진 *FileName*이 인용 부호로 묶여 있으면 인용 부호를 제거하고 묶여 있지 않은 *FileName*이 반환됩니다. *TextBetween* 함수가 구분 기호를 찾지 못하면 null을 반환하며, **Coalesce**는 이를 거부하고 대신 원래 *FileName*을 반환합니다.

if

if 함수는 함수에 제공된 조건이 True로 평가되는지, False로 평가되는지에 따른 값을 반환합니다.

구문:

```
if(condition , then [, else])
```

인수

인수	설명
condition	논리적으로 해석되는 표현식입니다.
then	어떤 유형이든 될 수 있는 표현식입니다. *condition*이 True인 경우 if 함수는 *then* 표현식의 값을 반환합니다.
else	어떤 유형이든 될 수 있는 표현식입니다. *condition*이 False인 경우 if 함수는 *else* 표현식의 값을 반환합니다. 이 매개 변수는 선택 사항입니다. *condition*이 False인 경우, else를 지정하지 않으면 NULL이 반환됩니다.

예

예	결과
`if( Amount>= 0, 'OK', 'Alarm' )`	이 표현식은 Amount가 양수(0 이상)인지 테스트하여 양수인 경우 'OK'를 반환합니다. Amount가 0 미만이면 'Alarm'이 반환됩니다.

예 - if를 사용한 로드 스크립트

예: 로드 스크립트

로드 스크립트

If는 변수를 포함하여 다른 메서드 및 개체와 함께 로드 스크립트에서 사용할 수 있습니다. 예를 들어 변수 *threshold*를 설정하고 해당 임계값에 기반한 데이터 모델에서 필드를 포함시키려면 다음을 수행하면 됩니다.

데이터 로드 편집기에서 새 탭을 만든 후 다음 데이터를 인라인 로드로 로드합니다. 결과를 보려면 Qlik Sense에서 아래 테이블을 만듭니다.

```
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;
```

결과

로드 스크립트에서 *if* 함수를 사용하여 나온 출력을 보여 주는 Qlik Sense 테이블입니다.

transaction_id	임계값과 비교
3750	Less than 100
3751	Greater than 100
3752	Less than 100
3753	Greater than 100
3754	Greater than 100
3756	Less than 100
3757	Greater than 100

예 - if를 사용한 차트 표현식

예: 차트 표현식

차트 표현식 1

로드 스크립트

데이터 로드 편집기에서 새 탭을 만든 후 다음 데이터를 인라인 로드로 로드합니다. 데이터를 로드한 후 아래의 차트 표현식 예를 Qlik Sense 테이블에 만듭니다.

```
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];
```

차트 표현식에서 *if* 함수의 예를 보여 주는 Qlik Sense 테이블입니다.

날짜	위치	Incidents	if(Incidents>=10, 'Critical', 'Ok')	if(Incidents>=10, 'Critical', If(Incidents>=1 and Incidents<10, 'Warning', 'Ok'))
1/3/2016	Beijing	0	Ok	Ok
1/3/2016	Boston	12	Critical	Critical
1/3/2016	Stockholm	3	Ok	Warning
1/3/2016	Toronto	0	Ok	Ok
1/4/2016	Beijing	0	Ok	Ok
1/4/2016	Boston	8	Ok	경고

차트 표현식 2

새 앱에서 데이터 로드 편집기의 새 탭에 다음 스크립트를 추가한 후 데이터를 로드합니다. 그런 다음 아래 차트 표현식으로 테이블을 만들 수 있습니다.

```
SET FirstWeekDay=0;
Load
Date(MakeDate(2022)+RecNo()-1) as Date
Autogenerate 14;
```

차트 표현식에서 *if* 함수의 예를 보여 주는 Qlik Sense 테이블입니다.

Date	WeekDay(Date)	If(WeekDay (Date)>=5,'WeekEnd','Normal Day')
1/1/2022	Sat	WeekEnd
1/2/2022	Sun	WeekEnd
1/3/2022	Mon	Normal Day
1/4/2022	Tue	Normal Day
1/5/2022	Wed	Normal Day
1/6/2022	Thu	Normal Day
1/7/2022	Fri	Normal Day
1/8/2022	Sat	WeekEnd
1/9/2022	Sun	WeekEnd
1/10/2022	Mon	Normal Day

Date	WeekDay(Date)	If(WeekDay (Date)>=5,'WeekEnd','Normal Day')
1/11/2022	Tue	Normal Day
1/12/2022	Wed	Normal Day
1/13/2022	Thu	Normal Day
1/14/2022	Fri	Normal Day

match

match 함수는 첫 번째 파라메타를 이후의 모든 파라메타와 비교하여 일치하는 표현식의 수 위치를 반환합니다. 대/소문자가 구분됩니다.

구문:

```
match( str, expr1 [ , expr2,...exprN ])
```

*대/소문자를 구분하지 않는 비교를 사용하려면 **mixmatch** 함수를 사용하십시오. 대/소문자를 구분하지 않는 비교 및 와일드카드를 사용하려면 **wildmatch** 함수를 사용하십시오.*

예: match를 사용한 로드 스크립트

예: 로드 스크립트

로드 스크립트

match를 사용하여 데이터 하위 집합을 로드할 수 있습니다. 예를 들어 함수의 표현식에 대해 숫자 값을 반환할 수 있습니다. 그런 다음 숫자 값에 따라 로드된 데이터를 제한할 수 있습니다. 일치하는 값이 없는 경우 Match가 0을 반환합니다. 그러므로 이 예에서 일치되지 않는 모든 표현식은 0을 반환하고 WHERE 문에 따라 데이터 로드에서 제외됩니다.

데이터 로드 편집기에서 새 탭을 만든 후 다음 데이터를 인라인 로드로 로드합니다. 결과를 보려면 Qlik Sense에서 아래 테이블을 만듭니다.

```
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;
```

결과

로드 스크립트에서 match 함수를 사용하여 나온 출력을 보여 주는 Qlik Sense 테이블

Color Code Blue and Black	Customer
Black	203521
Black	3036491
Blue	2038593

예 - match를 사용한 차트 표현식

예: 차트 표현식

차트 표현식 1

로드 스크립트

데이터 로드 편집기에서 새 탭을 만든 후 다음 데이터를 인라인 로드로 로드합니다. 데이터를 로드한 후 아래의 차트 표현식 예를 Qlik Sense 테이블에 만듭니다.

```
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];
```

아래 테이블의 첫 번째 표현식은 'Stockholm'이 **match** 함수의 표현식 목록에 포함되어 있지 않기 때문에 Stockholm에 대해 0을 반환합니다. 또한 **match** 비교는 대/소문자를 구분하기 때문에 'Zurich'의 경우 0을 반환합니다.

차트 표현식에서 Qlik Sense 함수의 예를 보여 주는 *match* 테이블입니다.

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

차트 표현식 2

사용자 지정 정렬 표현식을 수행하는 데 match를 사용할 수 있습니다.

기본적으로 열은 데이터에 따라 숫자순 또는 사전순으로 정렬됩니다.

기본 정렬 순서의 예를 보여 주는 Qlik Sense 테이블

Cities
Beijing
Boston
Stockholm
Toronto
zurich

순서를 변경하려면 다음을 수행합니다.

1. **속성** 패널에서 차트에 대해 **정렬** 섹션을 엽니다.
2. 사용자 지정 정렬을 수행하려는 열에 대해 자동 정렬을 끕니다.
3. **숫자순 정렬** 및 **사전순 정렬**을 선택 취소합니다.
4. **정렬 표현식**을 선택하고 다음과 유사한 표현식을 입력합니다.
   `=match( Cities, 'Toronto','Boston','Beijing','Stockholm','zurich')`
   Cities 열의 정렬 순서가 변경됩니다.

match 함수를 사용하여 정렬 순서의 변경 예를 보여 주는 Qlik Sense 테이블

Cities
Toronto
Boston
Beijing
Stockholm
zurich

또한 반환되는 숫자 값을 볼 수도 있습니다.

match 함수에서 반환되는 숫자 값의 예를 보여 주는 Qlik Sense 테이블

도시	**Cities & ' - ' & match (Cities, 'Toronto','Boston', 'Beijing','Stockholm','zurich')**
Toronto	Toronto - 1
Boston	Boston - 2
Beijing	Beijing - 3
Stockholm	Stockholm - 4
zurich	zurich - 5

mixmatch

mixmatch 함수는 첫 번째 파라메타를 이후의 모든 파라메타와 비교하여 일치하는 표현식의 수 위치를 반환합니다. 대/소문자는 구분되지 않습니다.

구문:

```
mixmatch( str, expr1 [ , expr2,...exprN ])
```

대신 대/소문자를 구분하는 비교를 사용하려면 **match** 함수를 사용하십시오. 대/소문자를 구분하지 않는 비교 및 와일드카드를 사용하려면 **wildmatch** 함수를 사용하십시오.

예 - mixmatch를 사용한 로드 스크립트

예: 로드 스크립트

로드 스크립트

mixmatch를 사용하여 데이터 하위 집합을 로드할 수 있습니다. 예를 들어 함수의 표현식에 대해 숫자 값을 반환할 수 있습니다. 그런 다음 숫자 값에 따라 로드된 데이터를 제한할 수 있습니다. 일치하는 값이 없는 경우 Mixmatch가 0을 반환합니다. 그러므로 이 예에서 일치되지 않는 모든 표현식은 0을 반환하고 WHERE 문에 따라 데이터 로드에서 제외됩니다.

데이터 로드 편집기에서 새 탭을 만든 후 다음 데이터를 인라인 로드로 로드합니다. 결과를 보려면 Qlik Sense에서 아래 테이블을 만듭니다.

```
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;
```

결과

로드 스크립트에서 mixmatch 함수를 사용하여 나온 출력을 보여 주는 Qlik Sense 테이블입니다.

Color Code Black, Blue, blue	Customer
Black	203521
Black	3036491
Blue	2038593
Blue	5646471

예 - mixmatch를 사용한 차트 표현식

예: 차트 표현식
데이터 로드 편집기에서 새 탭을 만든 후 다음 데이터를 인라인 로드로 로드합니다. 데이터를 로드한 후 아래의 차트 표현식 예를 Qlik Sense 테이블에 만듭니다.

차트 표현식 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];
```

아래 테이블의 첫 번째 표현식은 'Stockholm'이 **mixmatch** 함수의 표현식 목록에 포함되어 있지 않기 때문에 Stockholm에 대해 0을 반환합니다. **mixmatch** 비교는 대/소문자를 구분하지 못하기 때문에 'Zurich'의 경우 4를 반환합니다.

차트 표현식에서 *mixmatch* 함수의 예를 보여 주는 Qlik Sense 테이블

Cities	mixmatch(Cities,'Toronto','Boston','Beijing','Zurich')	mixmatch(Cities,'Toronto','Boston','Beijing','Stockholm','Zurich')
Beijing	3	3
Boston	2	2

Cities	mixmatch(Cities,'Toronto','Boston','Beijing','Zurich')	mixmatch(Cities,'Toronto','Boston','Beijing','Stockholm','Zurich')
Stockholm	0	4
Toronto	1	1
zurich	4	5

차트 표현식 2

사용자 지정 정렬 표현식을 수행하는 데 mixmatch를 사용할 수 있습니다.

기본적으로 열은 데이터에 따라 사전순 또는 숫자순으로 정렬됩니다.

기본 정렬 순서의 예를 보여 주는 Qlik Sense 테이블

Cities
Beijing
Boston
Stockholm
Toronto
zurich

순서를 변경하려면 다음을 수행합니다.

1. **속성** 패널에서 차트에 대해 **정렬** 섹션을 엽니다.
2. 사용자 지정 정렬을 수행하려는 열에 대해 자동 정렬을 끕니다.
3. **숫자순 정렬** 및 **사전순 정렬**을 선택 취소합니다.
4. **정렬 표현식**을 선택하고 다음 표현식을 입력합니다.
   `=mixmatch( Cities, 'Toronto','Boston','Beijing','Stockholm','Zurich')`
   Cities 열의 정렬 순서가 변경됩니다.

mixmatch 함수를 사용하여 정렬 순서의 변경 예를 보여 주는 Qlik Sense 테이블입니다.

Cities
Toronto
Boston
Beijing
Stockholm
zurich

또한 반환되는 숫자 값을 볼 수도 있습니다.

mixmatch 함수에서 반환되는 숫자 값의 예를 보여 주는 Qlik Sense 테이블입니다.

도시	Cities & ' - ' & mixmatch (Cities, 'Toronto','Boston', 'Beijing','Stockholm','Zurich')
Toronto	Toronto - 1
Boston	Boston - 2
Beijing	Beijing - 3
Stockholm	Stockholm - 4
zurich	zurich - 5

pick

pick 함수는 목록 내 *n*번째 표현식을 반환합니다.

구문:

```
pick(n, expr1[ , expr2,...exprN])
```

인수:

인수

인수	설명
n	*n*은 1과 N 사이의 정수입니다.

예

예	결과
`pick( N, 'A','B',4, 6 )`	'B' = N인 경우 2를 반환합니다. 4 = N인 경우 3를 반환합니다.

wildmatch

wildmatch 함수는 첫 번째 파라메타를 이후의 모든 파라메타와 비교하여 일치하는 표현식의 수를 반환합니다. 비교 문자열에서 와일드카드 문자(***** 및 **?**)의 사용을 허용합니다. *****는 문자의 순서와 일치합니다. **?**는 단일 문자와 일치합니다. 대/소문자는 구분되지 않습니다.

구문:

```
wildmatch( str, expr1 [ , expr2,...exprN ])
```

와일드카드 없이 비교를 사용하려면 **match** 또는 **mixmatch** 함수를 사용하십시오.

예: wildmatch를 사용한 로드 스크립트

예: 로드 스크립트

로드 스크립트

wildmatch를 사용하여 데이터 하위 집합을 로드할 수 있습니다. 예를 들어 함수의 표현식에 대해 숫자 값을 반환할 수 있습니다. 그런 다음 숫자 값에 따라 로드된 데이터를 제한할 수 있습니다. 일치하는 값이 없는 경우 Wildmatch가 0을 반환합니다. 그러므로 이 예에서 일치되지 않는 모든 표현식은 0을 반환하고 WHERE 문에 따라 데이터 로드에서 제외됩니다.

데이터 로드 편집기에서 새 탭을 만든 후 다음 데이터를 인라인 로드로 로드합니다. 결과를 보려면 Qlik Sense에서 아래 테이블을 만듭니다.

```
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;
```

결과

로드 스크립트에서 *wildmatch* 함수를 사용하여 나온 출력을 보여 주는 Qlik Sense 테이블

Color Code Black, Blue, blue	Customer
Black	203521
Black	3036491
Blue	2038593
Blue	5646471
Red	049681
Red	2038593

예: wildmatch를 사용한 차트 표현식

예: 차트 표현식

차트 표현식 1

데이터 로드 편집기에서 새 탭을 만든 후 다음 데이터를 인라인 로드로 로드합니다. 데이터를 로드한 후 아래의 차트 표현식 예를 Qlik Sense 테이블에 만듭니다.

```
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];
```

아래 테이블의 첫 번째 표현식은 'Stockholm'이 **wildmatch** 함수의 표현식 목록에 포함되어 있지 않기 때문에 Stockholm에 대해 0을 반환합니다. 또한 ?는 단일 문자에만 일치하기 때문에 'Boston'의 경우 0을 반환합니다.

차트 표현식에서 *wildmatch* 함수의 예를 보여 주는 Qlik Sense 테이블

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

차트 표현식 2

사용자 지정 정렬 표현식을 수행하는 데 wildmatch를 사용할 수 있습니다.

기본적으로 열은 데이터에 따라 숫자순 또는 사전순으로 정렬됩니다.

기본 정렬 순서의 예를 보여 주는 Qlik Sense 테이블

Cities
Beijing
Boston
Stockholm
Toronto
zurich

순서를 변경하려면 다음을 수행합니다.

1. **속성** 패널에서 차트에 대해 **정렬** 섹션을 엽니다.
2. 사용자 지정 정렬을 수행하려는 열에 대해 자동 정렬을 끕니다.
3. **숫자순 정렬** 및 **사전순 정렬**을 선택 취소합니다.
4. **정렬 표현식**을 선택하고 다음과 유사한 표현식을 입력합니다.
   `=wildmatch( Cities, 'Tor*','???ton','Beijing','Stockholm','*urich')`
   Cities 열의 정렬 순서가 변경됩니다.

wildmatch 함수를 사용하여 정렬 순서의 변경 예를 보여 주는 Qlik Sense 테이블입니다.

Cities
Toronto
Boston
Beijing
Stockholm
zurich

또한 반환되는 숫자 값을 볼 수도 있습니다.

wildmatch 함수에서 반환되는 숫자 값의 예를 보여 주는 Qlik Sense 테이블

도시	**Cities & ' - ' & wildmatch (Cities, 'Tor*','???ton','Beijing','Stockholm','*urich')**
Toronto	Toronto - 1
Boston	Boston - 2
Beijing	Beijing - 3
Stockholm	Stockholm - 4
zurich	zurich - 5

8.6 카운터 함수

이 섹션에서는 데이터 로드 스크립트에서 **LOAD** 문을 평가하는 동안 사용하는 레코드 카운터와 관련된 함수를 설명합니다. 차트 표현식에서는 **RowNo()** 함수만 사용할 수 있습니다.

일부 카운터 함수에는 매개 변수가 없지만 후행 괄호는 여전히 필요합니다.

카운터 함수 개요

각 함수는 개요가 끝난 후에 더 자세히 설명합니다. 구문에서 함수 이름을 클릭하여 해당 함수에 대한 상세 설명에 즉시 액세스할 수도 있습니다.

autonumber

이 스크립트 함수는 스크립트 실행 중에 발견된 *expression*의 평가된 각 고유 값에 해당하는 고유 정수 값을 반환합니다. 이 함수는 복잡한 키의 간단한 메모리 표현을 만드는 경우 등에 사용할 수 있습니다.

```
autonumber (expression[ , AutoID])
```

autonumberhash128

이 스크립트 함수는 복합 입력 표현식 값의 128비트 해시를 계산하고, 스크립트 실행 중에 발견된 각각의 고유 해시 값에 해당하는 고유 정수 값을 반환합니다. 이 함수는 복잡한 키의 간단한 메모리 표현을 만드는 경우 등에 사용할 수 있습니다.

```
autonumberhash128 (expression {, expression})
```

autonumberhash256
이 스크립트 함수는 복합 입력 표현식 값의 256비트 해시를 계산하고, 스크립트 실행 중에 발견된 각각의 고유 해시 값에 해당하는 고유 정수 값을 반환합니다. 이 함수는 복잡한 키의 간단한 메모리 표현을 만드는 경우 등에 사용할 수 있습니다.

```
autonumberhash256 (expression {, expression})
```

IterNo
이 스크립트 함수는 단일 레코드가 **LOAD** 문에서 **while** 절로 평가된 시기를 나타내는 정수를 반환합니다. 첫 번째 반복은 1번입니다. **IterNo** 함수는 **while** 절과 함께 사용할 경우에만 유효합니다.

```
IterNo ( )
```

RecNo
이 스크립트 함수는 현재 테이블의 현재 읽은 행의 수에 해당하는 정수를 반환합니다. 첫 번째 레코드는 1번입니다.

```
RecNo ( )
```

RowNo - script function
이 함수는 결과 Qlik Sense 내부 테이블에서 현재 행의 위치에 해당하는 정수를 반환합니다. 첫 번째 행은 1번입니다.

```
RowNo ( )
```

RowNo - chart function
RowNo()는 테이블의 현재 열 세그먼트 내에 있는 현재 행의 수를 반환합니다. 비트맵 차트의 경우, **RowNo()**는 해당 차트의 일반표 해당 부분 내에 있는 현재 행 번호를 반환합니다.

```
RowNo - 차트 함수([TOTAL])
```

autonumber

이 스크립트 함수는 스크립트 실행 중에 발견된 *expression*의 평가된 각 고유 값에 해당하는 고유 정수 값을 반환합니다. 이 함수는 복잡한 키의 간단한 메모리 표현을 만드는 경우 등에 사용할 수 있습니다.

테이블을 읽는 순서에 따라 정수가 생성되므로 동일한 데이터 로드 시 생성된 ***autonumber*** *키만 연결할 수 있습니다. 데이터 로드 간에 영구적인 키를 사용해야 하는 경우, 소스 데이터 정렬과 관계없이* ***hash128, hash160*** *또는* ***hash256*** *함수를 사용해야 합니다.*

구문:

```
autonumber(expression[ , AutoID])
```

인수:

인수	설명
AutoID	스크립트 내의 여러 키에 대해 **autonumber** 함수를 사용할 때 복수의 카운터 인스턴스를 만들려면 선택적 매개 변수인 *AutoID*를 사용하여 각 카운터에 이름을 지정할 수 있습니다.

복합 키 만들기

이 예에서는 메모리를 절약하기 위해 **autonumber** 함수를 사용하여 복합 키를 만듭니다. 예는 데모용으로 간략하게 나와 있지만 다량의 행이 포함된 테이블에서는 의미가 있습니다.

데이터 예

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

소스 데이터는 인라인 데이터를 사용하여 로드됩니다. 그다음, Region, Year 및 Month 필드에서 복합 키를 만드는 선행 LOAD를 추가합니다.

```
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
];
```

결과 테이블은 다음과 같습니다.

결과 테이블

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

이 예에서는 다른 테이블에 연결해야 하는 경우 'North2014May' 문자열 대신 RYMkey(예: 1)를 참조할 수 있습니다.

이제 비용에 대한 소스 테이블을 비슷한 방법으로 로드합니다. Region, Year 및 Month 필드는 가상 키가 만들어지지 않도록 하기 위해 선행 LOAD에서 제외되며, **autonumber** 함수를 사용하여 테이블을 연결하는 복합 키를 이미 만드는 중입니다.

```
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
];
```

이제 테이블 시각화를 시트에 추가하고, Region, Year 및 Month 필드와 더불어 Sum 측정값을 Sales 및 Costs에 대해 추가할 수 있습니다. 테이블은 다음과 같습니다.

결과 테이블

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

이 스크립트 함수는 복합 입력 표현식 값의 128비트 해시를 계산하고, 스크립트 실행 중에 발견된 각각의 고유 해시 값에 해당하는 고유 정수 값을 반환합니다. 이 함수는 복잡한 키의 간단한 메모리 표현을 만드는 경우 등에 사용할 수 있습니다.

테이블을 읽는 순서에 따라 정수가 생성되므로 동일한 데이터 로드 시 생성된 ***autonumberhash128*** *키만 연결할 수 있습니다. 데이터 로드 간에 영구적인 키를 사용해야 하는 경우, 소스 데이터 정렬과 관계없이* ***hash128, hash160*** *또는* ***hash256*** *함수를 사용해야 합니다.*

구문:

```
autonumberhash128(expression {, expression})
```

복합 키 만들기

이 예에서는 메모리를 절약하기 위해 **autonumberhash128** 함수를 사용하여 복합 키를 만듭니다. 예는 데모용으로 간략하게 나와 있지만 다량의 행이 포함된 테이블에서는 의미가 있습니다.

데이터 예

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

소스 데이터는 인라인 데이터를 사용하여 로드됩니다. 그다음, Region, Year 및 Month 필드에서 복합 키를 만드는 선행 LOAD를 추가합니다.

```
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
];
```

결과 테이블은 다음과 같습니다.

결과 테이블

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

이 예에서는 다른 테이블에 연결해야 하는 경우 'North2014May' 문자열 대신 RYMkey(예: 1)를 참조할 수 있습니다.

이제 비용에 대한 소스 테이블을 비슷한 방법으로 로드합니다. Region, Year 및 Month 필드는 가상 키가 만들어지지 않도록 하기 위해 선행 LOAD에서 제외되며, **autonumberhash128** 함수를 사용하여 테이블을 연결하는 복합 키를 이미 만드는 중입니다.

```
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
];
```

이제 테이블 시각화를 시트에 추가하고, Region, Year 및 Month 필드와 더불어 Sum 측정값을 Sales 및 Costs에 대해 추가할 수 있습니다. 테이블은 다음과 같습니다.

결과 테이블

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

이 스크립트 함수는 복합 입력 표현식 값의 256비트 해시를 계산하고, 스크립트 실행 중에 발견된 각각의 고유 해시 값에 해당하는 고유 정수 값을 반환합니다. 이 함수는 복잡한 키의 간단한 메모리 표현을 만드는 경우 등에 사용할 수 있습니다.

테이블을 읽는 순서에 따라 정수가 생성되므로 동일한 데이터 로드 시 생성된 ***autonumberhash256*** *키만 연결할 수 있습니다. 데이터 로드 간에 영구적인 키를 사용해야 하는 경우, 소스 데이터 정렬과 관계없이* ***hash128, hash160*** *또는* ***hash256*** *함수를 사용해야 합니다.*

구문:

```
autonumberhash256(expression {, expression})
```

복합 키 만들기

이 예에서는 메모리를 절약하기 위해 **autonumberhash256** 함수를 사용하여 복합 키를 만듭니다. 예는 데모용으로 간략하게 나와 있지만 다량의 행이 포함된 테이블에서는 의미가 있습니다.

예 테이블

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

소스 데이터는 인라인 데이터를 사용하여 로드됩니다. 그다음, Region, Year 및 Month 필드에서 복합 키를 만드는 선행 LOAD를 추가합니다.

```
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
];
```

결과 테이블은 다음과 같습니다.

결과 테이블

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

이 예에서는 다른 테이블에 연결해야 하는 경우 'North2014May' 문자열 대신 RYMkey(예: 1)를 참조할 수 있습니다.

이제 비용에 대한 소스 테이블을 비슷한 방법으로 로드합니다. Region, Year 및 Month 필드는 가상 키가 만들어지지 않도록 하기 위해 선행 LOAD에서 제외되며, **autonumberhash256** 함수를 사용하여 테이블을 연결하는 복합 키를 이미 만드는 중입니다.

```
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
];
```

이제 테이블 시각화를 시트에 추가하고, Region, Year 및 Month 필드와 더불어 Sum 측정값을 Sales 및 Costs에 대해 추가할 수 있습니다. 테이블은 다음과 같습니다.

결과 테이블

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

이 스크립트 함수는 단일 레코드가 **LOAD** 문에서 **while** 절로 평가된 시기를 나타내는 정수를 반환합니다. 첫 번째 반복은 1번입니다. **IterNo** 함수는 **while** 절과 함께 사용할 경우에만 유효합니다.

구문:

```
IterNo( )
```

예 및 결과:

```
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
];
```

이 **LOAD** 문은 **StartDate** 및 **EndDate**로 정의된 범위 내에서 날짜마다 레코드를 하나씩 생성합니다.

이 예에서 결과 테이블은 다음과 같이 표시됩니다.

결과 테이블

Day	Date
1	2014-01-22
2	2014-01-23
3	2014-01-24
4	2014-01-25
5	2014-01-26

RecNo

이 스크립트 함수는 현재 테이블의 현재 읽은 행의 수에 해당하는 정수를 반환합니다. 첫 번째 레코드는 1번입니다.

구문:

```
RecNo( )
```

RowNo( )는 결과 Qlik Sense 테이블의 행을 계수하는 반면 **RecNo( )**는 원시 데이터 테이블의 레코드를 계수하며 원시 데이터 테이블이 다른 테이블에 연결될 때 재설정됩니다.

데이터 로드 스크립트

원시 데이터 테이블 로드:

```
Tab1:
LOAD * INLINE
[A, B
1, aa
2,cc
3,ee];
```

```
Tab2:
LOAD * INLINE
[C, D
5, xx
4,yy
6,zz];
```

선택한 행의 레코드 및 행 번호 로드:

```
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;
```

결과 Qlik Sense 내부 테이블:

결과 테이블

A	B	RecNo( )	RowNo( )
1	aa	1	1
3	ee	3	2
4	yy	2	3
6	zz	3	4

RowNo

이 함수는 결과 Qlik Sense 내부 테이블에서 현재 행의 위치에 해당하는 정수를 반환합니다. 첫 번째 행은 1번입니다.

구문:

```
RowNo( [TOTAL])
```

원시 데이터 테이블 내의 레코드를 계수하는 **RecNo( )**와 달리, **RowNo( )** 함수는 **where** 절로 제외된 레코드를 계수하지 않으며 원시 데이터 테이블이 다른 테이블에 연결된 경우에도 초기화되지 않습니다.

*선행 LOAD를 사용할 경우, 즉 동일한 테이블에서 여러 개의 누적된 **LOAD** 문을 읽을 경우 최상위 **RowNo( )** 문에만 **LOAD** 를 사용할 수 있습니다. 후속 **LOAD** 문에서 **RowNo( )**를 사용할 경우 0이 반환됩니다.*

데이터 로드 스크립트

원시 데이터 테이블 로드:

```
Tab1:
LOAD * INLINE
[A, B
1, aa
2,cc
3,ee];

Tab2:
LOAD * INLINE
[C, D
5, xx
4,yy
6,zz];
```

선택한 행의 레코드 및 행 번호 로드:

```
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;
```

결과 Qlik Sense 내부 테이블:

결과 테이블

A	B	RecNo( )	RowNo( )
1	aa	1	1
3	ee	3	2
4	yy	2	3
6	zz	3	4

RowNo - 차트 함수

RowNo()는 테이블의 현재 열 세그먼트 내에 있는 현재 행의 수를 반환합니다. 비트맵 차트의 경우, **RowNo()**는 해당 차트의 일반표 해당 부분 내에 있는 현재 행 번호를 반환합니다.

테이블 또는 테이블과 동등한 것에 여러 세로 차원이 있을 경우 현재 열 세그먼트에는 필드 간 정렬 순서에서 마지막 차원이 표시되는 열을 제외하고 모든 차원 열 내의 현재 행과 같은 값을 가진 행만 포함됩니다.

열 세그먼트

이 차트 함수가 차트의 표현식에서 사용되는 경우 차트의 y 값에 대한 정렬 또는 테이블의 표현식 열에 의한 정렬은 허용되지 않습니다. 따라서 해당 정렬 옵션이 자동으로 비활성화됩니다. 시각화 또는 테이블에서 이 차트 함수를 사용하면 시각화의 정렬이 이 함수에 대해 정렬된 입력으로 되돌아갑니다.

구문:

```
RowNo([TOTAL])
```

반환 데이터 유형: 정수

인수:

인수	설명
TOTAL	테이블이 1차원이거나 **TOTAL** 한정자를 인수로 사용하는 경우 현재 열 세그먼트는 항상 전체 열과 동등합니다.

예: RowNo를 사용한 차트 표현식

예 - 차트 표현식

로드 스크립트

데이터 로드 편집기에서 다음 데이터를 인라인 로드로 로드하여 아래 차트 표현식 예를 만듭니다.

```
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 '|');
```

차트 표현식

Customer 및 **UnitSales**를 차원으로 사용하여 Qlik Sense 시트에 테이블 시각화를 만듭니다. 각각 **Row in Segment**과 **Row Number**로 레이블이 지정된 측정값으로 RowNo( ) 및 RowNo(TOTAL)을 추가합니다. 테이블에 다음 표현식을 측정값으로 추가합니다.

```
If( RowNo( )=1, 0, UnitSales / Above( UnitSales ))
```

결과

Customer	UnitSales	Row in Segment	Row Number	If(RowNo( )=1, 0, UnitSales / Above(UnitSales))
Astrida	4	1	1	0
Astrida	9	2	2	2.25
Astrida	10	3	3	1.1111111111111
Betacab	2	1	4	0
Betacab	5	2	5	2.5
Betacab	25	3	6	5
Canutility	4	1	7	0
Canutility	8	2	8	2
Divadip	1	1	9	0
Divadip	4	2	10	4

설명

Row in Segment 열에 고객 Astrida에 대한 UnitSales 값이 포함된 열 세그먼트의 결과 1,2,3이 표시됩니다. 그러면 다음 열 세그먼트인 Betacab에서 다시 1부터 행 번호 지정이 시작됩니다.

RowNo()에 대한 TOTAL 인수로 인해 **Row Number** 열은 차원을 무시하고 테이블의 행을 계산합니다.

이 표현식은 각 열 세그먼트의 첫 번째 행에 대해 0을 반환하므로 열에 다음이 표시됩니다.

0, 2.25, 1.1111111, 0, 2.5, 5, 0, 2, 0 및 4.

관련 항목:

Above - 차트 함수 (page 1228)

8.7 날짜 및 시간 함수

Qlik Sense 날짜 및 시간 함수는 날짜와 시간 값을 변환하는 데 사용됩니다. 모든 함수는 데이터 로드 스크립트와 차트 표현식 모두에서 사용할 수 있습니다.

이 함수는 1899년 12월 30일 이후의 일 수와 동일한 날짜-시간 일련 번호를 기준으로 합니다. 정수 값은 날짜를 나타내고 소수 값은 해당 날짜의 시간을 나타냅니다.

Qlik Sense는 매개 변수의 숫자 값을 사용하므로 숫자는 날짜 또는 시간으로 서식이 지정되지 않았을 때도 매개 변수로 유효합니다. 매개 변수가 문자열인 경우와 같이 숫자 값에 해당하지 않는 경우 Qlik Sense는 날짜 및 시간 환경 변수에 따라 문자열을 해석하려고 시도합니다.

매개 변수에 사용된 시간 서식이 환경 변수로 설정된 서식과 일치하지 않을 경우 Qlik Sense에서 정확한 해석을 수행할 수 없게 됩니다. 이 문제를 해결하려면 설정을 변경하거나 해석 함수를 사용하십시오.

이 예의 각 함수에는 기본 시간 및 날짜 서식인 hh:mm:ss와 YYYY-MM-DD (ISO 8601)이 사용됩니다.

> *날짜 또는 시간 함수를 사용하여 타임스탬프를 처리할 때 날짜 또는 시간 함수에 지리적 위치가 포함되어 있지 않으면 Qlik Sense는 일광 절약 시간 매개 변수를 무시합니다.*
>
> *예를 들어 `ConvertToLocalTime( filetime('Time.qvd'), 'Paris')`은 일광 절약 시간 매개 변수를 사용하고 `ConvertToLocalTime(filetime('Time.qvd'), 'GMT-01:00')`은 일광 절약 시간 매개 변수를 사용하지 않습니다.*

날짜 및 시간 함수 개요

각 함수는 개요가 끝난 후에 더 자세히 설명합니다. 구문에서 함수 이름을 클릭하여 해당 함수에 대한 상세 설명에 즉시 액세스할 수도 있습니다.

시간의 정수 표현식

second
이 함수는 **expression**의 분위수가 표준 숫자 해석에 따라 시간으로 해석될 경우 초를 나타내는 정수를 반환합니다.

```
second (expression)
```

minute
이 함수는 **expression**의 분위수가 표준 숫자 해석에 따라 시간으로 해석될 경우 분을 나타내는 정수를 반환합니다.

```
minute (expression)
```

hour
이 함수는 **expression**의 분위수가 표준 숫자 해석에 따라 시간으로 해석될 경우 시간을 나타내는 정수를 반환합니다.

```
hour (expression)
```

day
이 함수는 **expression**의 분위수가 표준 숫자 해석에 따라 날짜로 해석될 경우 일을 나타내는 정수를 반환합니다.

```
day (expression)
```

week
이 함수는 ISO 8601에 따른 주 번호를 나타내는 정수를 반환합니다. 주 번호는 표준 숫자 해석에 따라 표현식의 날짜 해석을 통해 계산됩니다.

```
week (expression)
```

month
이 함수는 환경 변수 **MonthNames**로 정의되고 1~12 사이의 정수인 월 이름이 포함된 이중 값을 반환합니다. 월은 표준 숫자 해석에 따라 표현식의 날짜 해석을 통해 계산됩니다.

```
month (expression)
```

year
이 함수는 **expression**이 표준 숫자 해석에 따라 날짜로 해석될 경우 연도를 나타내는 정수를 반환합니다.

```
year (expression)
```

weekyear
이 함수는 환경 변수에 따라 해당 주차가 속한 연도를 반환합니다. 주차 범위는 1과 약 52 사이입니다.

```
weekyear (expression)
```

weekday
이 함수는 다음을 포함하는 이중 값을 반환합니다.

- 환경 변수 **DayNames**로 정의한 날짜 이름.
- 주의 명목상 이름(0~6)에 해당하는 0~6 사이의 정수.

```
weekday (date)
```

타임스탬프 함수

now
이 함수는 현재 시간의 타임스탬프를 반환합니다. 이 함수는 **TimeStamp** 시스템 변수 서식으로 값을 반환합니다. 기본 **timer_mode** 값은 1입니다.

```
now ([ timer_mode])
```

today
이 함수는 현재 날짜를 반환합니다. 이 함수는 DateFormat 시스템 변수 서식으로 값을 반환합니다.

```
today ([timer_mode])
```

LocalTime
이 함수는 지정된 표준 시간대에 대한 현재 시간의 타임스탬프를 반환합니다.

```
localtime ([timezone [, ignoreDST ]])
```

Make 함수

makedate
이 함수는 연도 **YYYY**, 월 **MM**, 일 **DD**에서 계산한 날짜를 반환합니다.

```
makedate (YYYY [ , MM [ , DD ] ])
```

makeweekdate
이 함수는 연도, 주차 및 요일에서 계산된 날짜를 반환합니다.

```
makeweekdate (YYYY [ , WW [ , D ] ])
```

maketime
이 함수는 시간 **hh**, 분 **mm**, 초 **ss**에서 계산한 시간을 반환합니다.

```
maketime (hh [ , mm [ , ss [ .fff ] ] ])
```

기타 날짜 함수

AddMonths
이 함수는 **startdate**를 기준으로 **n**개월 후의 날짜, 또는 **n**이 음수일 경우는 **startdate**를 기준으로 **n**개월 전의 날짜를 반환합니다.

```
addmonths (startdate, n , [ , mode])
```

AddYears
이 함수는 **startdate**를 기준으로 **n**년 후의 날짜, 또는 **n**이 음수일 경우는 **startdate**를 기준으로 **n**년 전의 날짜를 반환합니다.

```
addyears (startdate, n)
```

yeartodate
이 함수는 스크립트를 마지막으로 로드한 날짜의 연도 내에 입력 타임스탬프가 포함되는지 파악하고, 포함되는 경우 True, 포함되지 않는 경우 False를 반환합니다.

```
yeartodate (date [ , yearoffset [ , firstmonth [ , todaydate] ] ])
```

표준 시간대 함수

timezone
이 함수는 Qlik 엔진이 실행 중인 컴퓨터에 정의된 표준 시간대를 반환합니다.

```
timezone ( )
```

GMT
이 함수는 지역 설정에서 유추한 현재 Greenwich Mean Time을 반환합니다.

```
GMT ( )
```

UTC
현재 Coordinated Universal Time을 반환합니다.

```
UTC ( )
```

daylightsaving
Windows에 정의된 일광 절약 시간제 시간의 현재 조정 내용을 반환합니다.

```
daylightsaving ( )
```

converttolocaltime
UTC 또는 GMT 타임스탬프를 이중 값 형태의 현지 시간으로 변환합니다. place는 전 세계 여러 도시, 장소, 표준 시간대로 지정할 수 있습니다.

```
converttolocaltime (timestamp [, place [, ignore_dst=false]])
```

시간 설정 함수

setdateyear
이 함수는 **timestamp** 및 **year** 를 입력으로 사용하여 입력에 지정된 **year** 로 **timestamp** 를 업데이트합니다.

```
setdateyear  (timestamp, year)
```

setdateyearmonth
이 함수는 **timestamp**, **month** 및 **year** 를 입력으로 사용하여 입력에 지정된 **year**및 **month** 로 **timestamp** 를 업데이트합니다.

```
setdateyearmonth  (timestamp, year, month)
```

In... 함수

inyear
timestamp가 **base_date**를 포함하는 연도에 속할 경우 이 함수는 True를 반환합니다.

```
inyear (date, basedate , shift [, first_month_of_year = 1])
```

inyeartodate
이 함수는 **timestamp**가 **base_date**의 마지막 밀리초까지 포함하여 **base_date**를 포함한 연도의 일부에 속할 경우 True를 반환합니다.

```
inyeartodate  (date, basedate , shift [, first_month_of_year = 1])
```

inquarter
timestamp가 **base_date**를 포함하는 분기에 속할 경우 이 함수는 True를 반환합니다.

```
inquarter  (date, basedate , shift [, first_month_of_year = 1])
```

inquartertodate
이 함수는 **timestamp**가 **base_date**의 마지막 밀리초까지 포함하여 **base_date**를 포함한 분기의 일부에 속할 경우 True를 반환합니다.

```
inquartertodate (date, basedate , shift [, first_month_of_year = 1])
```

inmonth
timestamp가 **base_date**를 포함하는 월에 속할 경우 이 함수는 True를 반환합니다.

```
inmonth (date, basedate , shift)
```

inmonthtodate
이 함수는 **date**가 **basedate**의 마지막 밀리초까지 포함하여 **basedate**를 포함한 월의 일부에 속할 경우True를 반환합니다.

```
inmonthtodate (date, basedate , shift)
```

inmonths
이 함수는 타임스탬프가 기준 날짜와 같은 월, 2개월, 분기, 4개월 기간 또는 6개월 내에 속하는지 찾습니다. 또한 타임스탬프가 이전 기간 또는 다음 기간 내에 속하는지도 알아낼 수 있습니다.

```
inmonths (n, date, basedate , shift [, first_month_of_year = 1])
```

inmonthstodate
이 함수는 타임스탬프가 base_date의 마지막 밀리초를 포함하여 월, 2개월, 분기, 4개월 기간 또는 6개월 기간에 속하는지 여부를 찾습니다. 또한 타임스탬프가 이전 기간 또는 다음 기간 내에 속하는지도 알아낼 수 있습니다.

```
inmonthstodate (n, date, basedate , shift [, first_month_of_year = 1])
```

inweek
timestamp가 **base_date**를 포함하는 주에 속할 경우 이 함수는 True를 반환합니다.

```
inweek (date, basedate , shift [, weekstart])
```

inweektodate
이 함수는 **timestamp**가 **base_date**의 마지막 밀리초까지 포함하여 **base_date**를 포함한 주의 일부에 속할 경우 True를 반환합니다.

```
inweektodate (date, basedate , shift [, weekstart])
```

inlunarweek
이 함수는 **timestamp**가 **base_date**를 포함하는 음력 주 안에 있는지 확인합니다. Qlik Sense에서 음력 주는 1월 1일을 주의 첫 번째 날로 계산하여 정의되며, 연도의 마지막 주를 제외하고 각 주는 정확히 7일을 포함합니다.

```
inlunarweek (date, basedate , shift [, weekstart])
```

inlunarweektodate
이 함수는 **timestamp**가 **base_date**의 마지막 밀리초까지 포함하여 음력 주의 일부에 속하는지 확인합니다. Qlik Sense에서 음력 주는 1월 1일을 주의 첫 번째 날로 계산하여 정의되며, 연도의 마지막 주를 제외하고 정확히 7일이 포함됩니다.

```
inlunarweektodate  (date, basedate , shift [, weekstart])
```

inday
timestamp가 **base_timestamp**를 포함하는 날에 속할 경우 이 함수는 True를 반환합니다.

```
inday (timestamp, basetimestamp , shift [, daystart])
```

indaytotime
이 함수는 **timestamp**가 **base_timestamp**의 정확한 밀리초까지 포함하여 **base_timestamp**를 포함한 날의 일부에 속할 경우 True를 반환합니다.

```
indaytotime (timestamp, basetimestamp , shift [, daystart])
```

Start … end 함수

yearstart
이 함수는 **date**를 포함하는 연도의 첫 번째 날의 시작에 해당하는 타임스탬프를 반환합니다. 기본 출력 형식은 스크립트에 설정된 **DateFormat**입니다.

```
yearstart ( date [, shift = 0 [, first_month_of_year = 1]])
```

yearend
이 함수는 **date**를 포함하는 연도의 마지막 날의 마지막 밀리초의 타임스탬프에 해당하는 값을 반환합니다. 기본 출력 형식은 스크립트에 설정된 **DateFormat**입니다.

```
yearend ( date [, shift = 0 [, first_month_of_year = 1]])
```

yearname
이 함수는 **date**를 포함한 연도의 첫 번째 날의 첫 번째 밀리초의 타임스탬프에 해당하는 기본 숫자 값을 사용하여 네 자리 연도를 표시 값으로 반환합니다.

```
yearname (date [, shift = 0 [, first_month_of_year = 1]] )
```

quarterstart
이 함수는 **date**를 포함하는 분기의 첫 번째 밀리초의 타임스탬프에 해당하는 값을 반환합니다. 기본 출력 형식은 스크립트에 설정된 **DateFormat**입니다.

```
quarterstart (date [, shift = 0 [, first_month_of_year = 1]])
```

quarterend
이 함수는 **date**를 포함하는 분기의 마지막 밀리초의 타임스탬프에 해당하는 값을 반환합니다. 기본 출력 형식은 스크립트에 설정된 **DateFormat**입니다.

```
quarterend (date [, shift = 0 [, first_month_of_year = 1]])
```

quartername
이 함수는 해당 분기의 첫 번째 날의 첫 번째 밀리초의 타임스탬프에 해당하는 기본 숫자 값으로 분기의 월 (**MonthNames** 스크립트 변수에 따라 서식 지정) 및 연도를 보여주는 표시 값을 반환합니다.

```
quartername (date [, shift = 0 [, first_month_of_year = 1]])
```

monthstart
이 함수는 **date**를 포함하는 월의 첫 번째 날의 첫 번째 밀리초의 타임스탬프에 해당하는 값을 반환합니다. 기본 출력 형식은 스크립트에 설정된 **DateFormat**입니다.

```
monthstart (date [, shift = 0])
```

monthend
이 함수는 date를 포함하는 월의 마지막 날의 마지막 밀리초의 타임스탬프에 해당하는 값을 반환합니다. 기본 출력 형식은 스크립트에 설정된 DateFormat입니다.

```
monthend (date [, shift = 0])
```

monthname
이 함수는 해당 월의 첫 번째 날의 첫 번째 밀리초의 타임스탬프에 해당하는 기본 숫자 값으로 월 (**MonthNames** 스크립트 변수에 따라 서식 지정) 및 연도를 보여주는 표시 값을 반환합니다.

```
monthname (date [, shift = 0])
```

monthsstart
이 함수는 기준일을 포함하는 월, 2개월, 분기, 4개월 기간 또는 6개월의 첫 번째 밀리초의 타임스탬프에 해당하는 값을 반환합니다. 또한 이전 기간 또는 다음 기간에 대한 타임스탬프를 찾을 수도 있습니다.기본 출력 형식은 스크립트에 설정된 **DateFormat**입니다.

```
monthsstart (n, date [, shift = 0 [, first_month_of_year = 1]])
```

monthsend
이 함수는 기준일을 포함하는 월, 2개월, 분기, 4개월 기간 또는 6개월의 마지막 밀리초의 타임스탬프에 해당하는 값을 반환합니다. 또한 이전 기간 또는 다음 기간에 대한 타임스탬프를 찾을 수도 있습니다.

```
monthsend (n, date [, shift = 0 [, first_month_of_year = 1]])
```

monthsname
이 함수는 연도뿐 아니라 기간의 월 범위(**MonthNames** 스크립트 변수에 따라 서식 지정)를 나타내는 표시 값도 반환합니다. 기본 숫자 값은 기준일을 포함하는 월, 2개월, 분기, 4개월 기간 또는 6개월의 첫 번째 밀리초의 타임스탬프에 해당합니다.

```
monthsname (n, date [, shift = 0 [, first_month_of_year = 1]])
```

weekstart
이 함수는 **date**를 포함하는 캘린더 주의 첫 번째 날의 첫 번째 밀리초의 타임스탬프에 해당하는 값을 반환합니다. 기본 출력 형식은 스크립트에 설정된 **DateFormat**입니다.

```
weekstart (date [, shift = 0 [,weekoffset = 0]])
```

weekend
이 함수는 **date**를 포함하는 캘린더 주의 마지막 날 마지막 밀리초의 타임스탬프에 해당하는 값을 반환합니다. 기본 출력 형식은 스크립트에 설정된 **DateFormat**입니다.

```
weekend (date [, shift = 0 [,weekoffset = 0]])
```

weekname
이 함수는 **date**를 포함한 주의 첫 번째 날의 첫 번째 밀리초의 타임스탬프에 해당하는 기본 숫자 값으로 연도 및 주차를 보여주는 값을 반환합니다.

```
weekname (date [, shift = 0 [,weekoffset = 0]])
```

lunarweekstart
이 함수는 **date**를 포함하는 음력 주의 첫 번째 날의 첫 번째 밀리초의 타임스탬프에 해당하는 값을 반환합니다. Qlik Sense에서 음력 주는 1월 1일을 주의 첫 번째 날로 계산하여 정의되며, 연도의 마지막 주를 제외하고 정확히 7일이 포함됩니다.

```
lunarweekstart (date [, shift = 0 [,weekoffset = 0]])
```

lunarweekend
이 함수는 **date**를 포함하는 음력 주의 마지막 날의 마지막 밀리초의 타임스탬프에 해당하는 값을 반환합니다. Qlik Sense에서 음력 주는 1월 1일을 주의 첫 번째 날로 계산하여 정의되며, 연도의 마지막 주를 제외하고 정확히 7일이 포함됩니다.

```
lunarweekend (date [, shift = 0 [,weekoffset = 0]])
```

lunarweekname
이 함수는 **date**를 포함한 음력 주의 첫 번째 날의 첫 번째 밀리초의 타임스탬프에 해당하는 음력 주차와 연도를 보여주는 표시 값을 반환합니다. Qlik Sense에서 음력 주는 1월 1일을 주의 첫 번째 날로 계산하여 정의되며, 연도의 마지막 주를 제외하고 정확히 7일이 포함됩니다.

```
lunarweekname (date [, shift = 0 [,weekoffset = 0]])
```

daystart
이 함수는 **time** 인수에 포함된 날의 첫 번째 밀리초가 있는 타임스탬프에 해당하는 값을 반환합니다. 기본 출력 형식은 스크립트에 설정된 **TimestampFormat**입니다.

```
daystart (timestamp [, shift = 0 [, dayoffset = 0]])
```

dayend
이 함수는 **time**에 포함된 날의 마지막 밀리초의 타임스탬프에 해당하는 값을 반환합니다. 기본 출력 형식은 스크립트에 설정된 **TimestampFormat**입니다.

```
dayend (timestamp [, shift = 0 [, dayoffset = 0]])
```

dayname
이 함수는 **time**을 포함한 날의 첫 번째 밀리초의 타임스탬프에 해당하는 기본 숫자 값으로 날짜를 표시하는 값을 반환합니다.

```
dayname (timestamp [, shift = 0 [, dayoffset = 0]])
```

날짜 계수 함수

age
age 함수는 **date_of_birth**에 태어난 사람이 **timestamp**의 시간에 (만으로) 몇 살인지 반환합니다.

```
age (timestamp, date_of_birth)
```

networkdays
networkdays 함수는 선택적으로 나열된 **holiday**를 고려하여 **start_date**(포함)와 **end_date**(포함) 사이의 근무일(월요일 ~ 금요일)의 수를 반환합니다.

```
networkdays  (start:date, end_date {, holiday})
```

firstworkdate
firstworkdate 함수는 선택적으로 나열된 공휴일을 고려하여 **no_of_workdays**(월요일 ~ 금요일)가 **end_date** 이전에 끝나게 되는 가장 최근의 시작 날짜를 반환합니다. **end_date** 및 **holiday**은 유효한 날짜 또는 타임스태프여야 합니다.

```
firstworkdate (end_date, no_of_workdays {, holiday} )
```

lastworkdate
lastworkdate 함수는 선택적으로 나열된 **holiday**를 고려하여 **start_date**에 시작하는 경우 **no_of_workdays**(월요일-금요일)가 끝나는 가장 빠른 끝 날짜를 반환합니다. **start_date** 및 **holiday**는 유효한 날짜 또는 타임스탬프여야 합니다.

```
lastworkdate (start_date, no_of_workdays {, holiday})
```

daynumberofyear
이 함수는 타임스탬프가 속하는 연도의 일수를 계산합니다. 해당 연도의 첫 날의 첫 번째 밀리초로부터 계산되지만 첫 번째 월은 오프셋 지정할 수 있습니다.

```
daynumberofyear (date[,firstmonth])
```

daynumberofquarter
이 함수는 타임스탬프가 속하는 분기의 일수를 계산합니다. 마스터 캘린더를 만들 때 사용하는 함수입니다.

```
daynumberofquarter (date[,firstmonth])
```

addmonths

이 함수는 **startdate**를 기준으로 **n**개월 후의 날짜, 또는 **n**이 음수일 경우는 **startdate**를 기준으로 **n**개월 전의 날짜를 반환합니다.

구문:

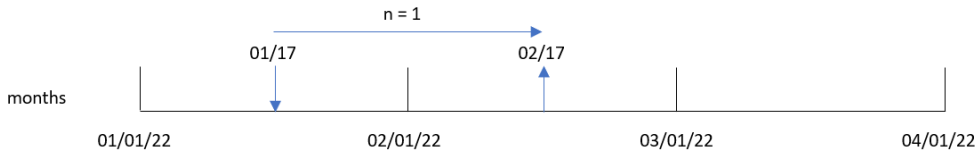
```
AddMonths(startdate, n , [ , mode])
```

반환 데이터 유형: dual

`addmonths()` 함수는 `startdate`에서 정의된 개월 수(`n`)를 더하거나 빼서 결과 날짜를 반환합니다.

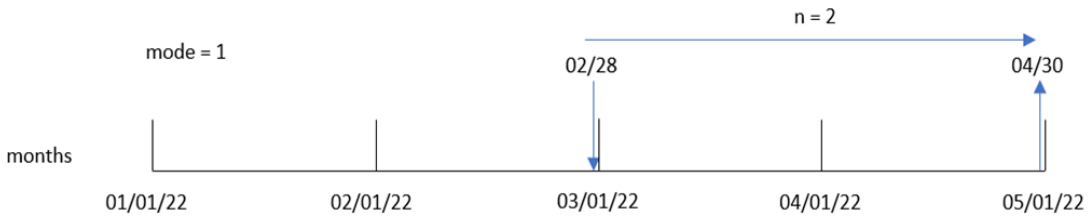
mode 인수는 해당 월 28일의 startdate 값 또는 그 이후의 값에 영향을 미칩니다. mode 인수를 1로 설정하면 addmonths() 함수는 월말까지의 상대적 기간이 동일한 날짜를 startdate로 반환합니다.

addmonths() 함수의 다이어그램 예



예를 들어, 2월 28일은 월의 마지막 날입니다. mode가 1인 addmonths() 함수를 사용하여 2개월 후 날짜를 반환하면 이 함수는 4월의 마지막 날짜인 4월 30일을 반환합니다.

mode=1인 addmonths() 함수의 다이어그램 예



인수

인수	설명
startdate	시작 날짜를 타임스탬프로 지정합니다(예: '2012-10-12').
n	월 수를 양의 정수 또는 음의 정수로 지정합니다.
mode	해당 월이 월 시작 날짜를 기준으로 추가될지 또는 끝 날짜를 기준으로 추가될지 지정합니다. 기본 모드는 0이며, 월의 시작 날짜를 기준으로 추가됩니다. 월의 끝 날짜를 기준으로 추가되는 경우 모드를 1로 설정합니다. 모드가 1로 설정되고 입력 날짜가 28이상이면 함수가 시작 날짜에서 월의 끝 날짜까지 며칠이 남았는지 확인합니다. 월의 끝 날짜까지 남아 있는 같은 날짜 수가 반환되는 날짜에 설정됩니다.

사용 시기

addmonths() 함수는 일반적으로 일정 기간의 주어진 개월 수 전후의 날짜를 찾는 표현식에서 사용됩니다.

예를 들어, addmonths() 함수를 사용하여 전화 계약의 종료 날짜를 식별할 수 있습니다.

함수 예

예	결과
addmonths ('01/29/2003' ,3)	'04/29/2003'을 반환합니다.
addmonths ('01/29/2003',3,0)	'04/29/2003'을 반환합니다.

예	결과
addmonths ('01/29/2003',3,1)	'04/28/2003'을 반환합니다.
addmonths ('01/29/2003',1,0)	'02/28/2003'을 반환합니다.
addmonths ('01/29/2003',1,1)	'02/26/2003'을 반환합니다.
addmonths ('02/28/2003',1,0)	'03/28/2003'을 반환합니다.
addmonths ('02/28/2003',1,1)	'03/31/2003'을 반환합니다.
addmonths ('01/29/2003',-3)	'10/29/2002'을 반환합니다.

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

예 1 – 추가 인수 없음

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Transactions라는 테이블에 로드되는 2020년과 2022년 사이의 트랜잭션 집합이 포함된 데이터 집합.
- DateFormat 시스템 변수(MM/DD/YYYY) 서식으로 제공된 날짜 필드.
- 트랜잭션이 발생한 후 2개월 동안의 날짜를 반환하는 필드 two_months_later 만들기.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- two_months_later

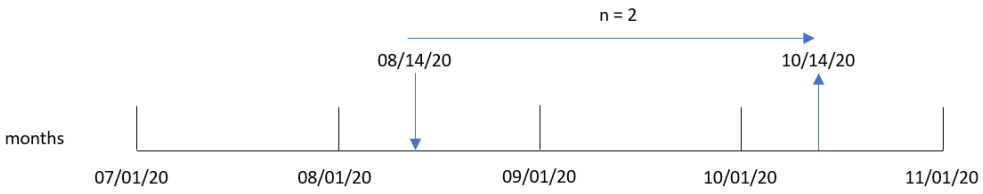
결과 테이블

date	two_months_later
01/10/2020	03/10/2020
02/28/2020	04/28/2020
04/09/2020	06/09/2020
04/16/2020	06/16/2020
05/21/2020	07/21/2020
08/14/2020	10/14/2020
10/07/2020	12/07/2020
12/05/2020	02/05/2021
01/22/2021	03/22/2021
02/03/2021	04/03/2021

date	two_months_later
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

'two_months_later' 필드는 addmonths() 함수를 사용하여 선행 LOAD 문에서 만들어집니다. 제공된 첫 번째 인수는 평가 중인 날짜를 식별합니다. 두 번째 인수는 startdate에서 더하거나 뺄 개월 수입니다. 이 경우 값 2가 제공됩니다.

addmonths() 함수 다이어그램, 추가 인수가 없는 예



트랜잭션 8193은 8월 14일에 발생했습니다. 따라서 addmonths() 함수는 two_months_later 필드에 대해 2020년 10월 14일을 반환합니다.

예 2 - 상대적인 월말

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Transactions라는 테이블에 로드되는 2022년 월말 트랜잭션 집합이 포함된 데이터 집합.
- DateFormat 시스템 변수(MM/DD/YYYY) 서식으로 제공된 날짜 필드.
- 트랜잭션이 발생하기 2개월 전의 상대적인 월말 날짜를 반환하는 필드 relative_two_months_prior 만들기.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

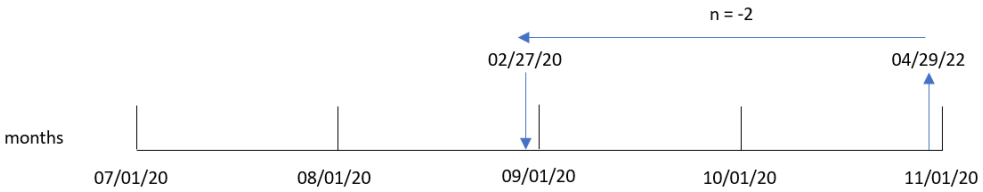
- date
- relative_two_months_prior

결과 테이블

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

relative_two_months_prior 필드는 addmonths() 함수를 사용하여 선행 LOAD 문에서 만들어집니다. 제공된 첫 번째 인수는 평가 중인 날짜를 식별합니다. 두 번째 인수는 startdate에서 더하거나 뺄 개월 수입니다. 이 경우 값 -2가 제공됩니다. 마지막 인수는 값이 1인 모드로, 함수가 28보다 크거나 같은 모든 날짜에 대해 상대적인 월말 날짜를 계산하도록 합니다.

addmonths() 함수 다이어그램(예: n=-2)



트랜잭션 8191은 2022년 4월 29일에 발생합니다. 초기에는 2개월 전의 월을 2월로 설정합니다. 그런 다음 함수의 세 번째 인수에서 모드가 1로 설정되고 일 값이 27일 이후이므로 이 함수는 상대적인 월말 값을 계산합니다. 이 함수는 29일이 4월의 두 번째 마지막 날임을 식별하므로 2월의 두 번째 마지막 날인 27일을 반환합니다.

예 3 – 차트 개체 예

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

이 로드 스크립트에는 첫 번째 예와 동일한 데이터 집합 및 시나리오가 포함되어 있습니다.

그러나 이 예에서는 변경되지 않은 데이터 집합이 응용 프로그램에 로드됩니다. 트랜잭션이 발생한 후 2개월 동안의 날짜를 반환하는 계산은 차트 개체에서 측정값으로 만들어집니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다. date.

다음 측정값을 만듭니다.

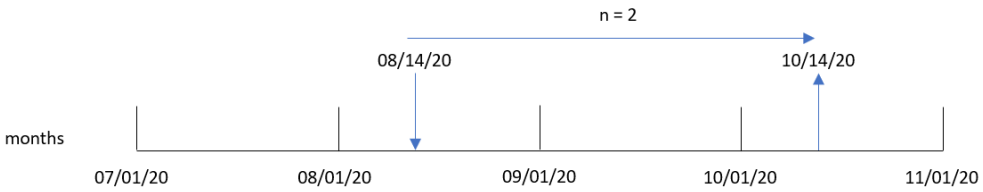
```
=addmonths(date,2)
```

결과 테이블

date	=addmonths(date,2)
01/10/2020	03/10/2020
02/28/2020	04/28/2020
04/09/2020	06/09/2020
04/16/2020	06/16/2020
05/21/2020	07/21/2020
08/14/2020	10/14/2020
10/07/2020	12/07/2020
12/05/2020	02/05/2021
01/22/2021	03/22/2021
02/03/2021	04/03/2021
03/17/2021	05/17/2021
04/23/2021	06/23/2021
05/04/2021	07/04/2021
06/30/2021	08/30/2021
07/26/2021	09/26/2021
12/27/2021	02/27/2022
02/02/2022	04/02/2022
02/26/2022	04/26/2022
03/07/2022	05/07/2022
03/11/2022	05/11/2022

two_months_later 측정값은 addmonths() 함수를 사용하여 차트 개체에 만들어집니다. 제공된 첫 번째 인수는 평가 중인 날짜를 식별합니다. 두 번째 인수는 startdate에서 더하거나 뺄 개월 수입니다. 이 경우 값 2가 제공됩니다.

addmonths() 함수의 다이어그램, 차트 개체 예



트랜잭션 8193은 8월 14일에 발생했습니다. 따라서 addmonths() 함수는 two_months_later 필드에 대해 2020년 10월 14일을 반환합니다.

예 4 – 시나리오

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Mobile_Plans라는 테이블에 로드되는 데이터 집합.
- 계약 ID, 시작 날짜, 계약 기간 및 월 사용료가 포함된 정보.

최종 사용자는 계약 ID별로 각 전화 계약의 종료 날짜를 표시하는 차트 개체를 원합니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- contract_id
- start_date
- contract_length

다음 측정값을 만들어 각 계약의 종료 날짜를 계산합니다.

```
=addmonths(start_date,contract_length, 0)
```

결과 테이블

contract_id	start_date	contract_length	=addmonths(start_date,contract_length,0)
8188	01/13/2020	18	07/13/2021
8189	02/26/2020	24	02/26/2022
8190	03/27/2020	36	03/27/2023
8191	04/16/2020	24	04/16/2022
8192	05/21/2020	24	05/21/2022
8193	08/14/2020	12	08/14/2021
8194	10/07/2020	18	04/07/2022
8195	12/05/2020	12	12/05/2021
8196	01/22/2021	12	01/22/2022
8197	02/03/2021	18	08/03/2022
8198	03/17/2021	24	03/17/2023
8199	04/23/2021	24	04/23/2023
8200	05/04/2021	12	05/04/2022
8201	06/30/2021	12	06/30/2022
8202	07/26/2021	12	07/26/2022
8203	12/27/2021	36	12/27/2024
8204	06/06/2022	24	06/06/2024

contract_id	start_date	contract_length	=addmonths(start_date,contract_length,0)
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

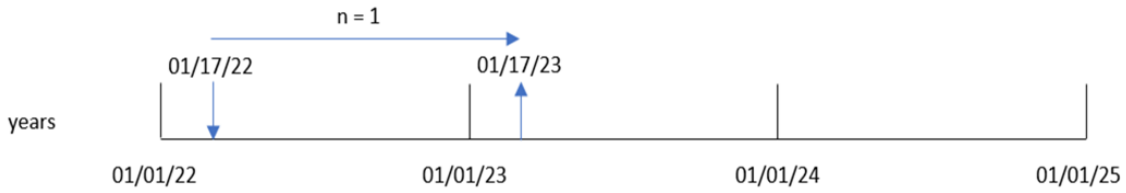
이 함수는 **startdate**를 기준으로 **n**년 후의 날짜, 또는 **n**이 음수일 경우는 **startdate**를 기준으로 **n**년 전의 날짜를 반환합니다.

구문:

```
AddYears(startdate, n)
```

반환 데이터 유형: dual

addyears() 함수의 다이어그램 예



addyears() 함수는 startdate에서 정의된 연도 수(n)를 더하거나 뺍니다. 그런 다음 결과 날짜를 반환합니다.

인수

인수	설명
startdate	시작 날짜를 타임스탬프로 지정합니다(예: '2012-10-12').
n	연도 수를 양의 정수 또는 음의 정수로 지정합니다.

함수 예

예	결과
addyears ('01/29/2010',3)	'01/29/2013'을 반환합니다.
addyears ('01/29/2010',-1)	'01/29/2009'을 반환합니다.

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

예 1 – 간단한 예

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- `Transactions`라는 테이블에 로드되는 2020년과 2022년 사이의 트랜잭션 집합이 포함된 데이터 집합.
- `DateFormat` 시스템 변수(MM/DD/YYYY) 서식으로 제공된 날짜 필드.
- 트랜잭션이 발생한 후 2년 동안의 날짜를 반환하는 필드 `two_years_later` 만들기.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

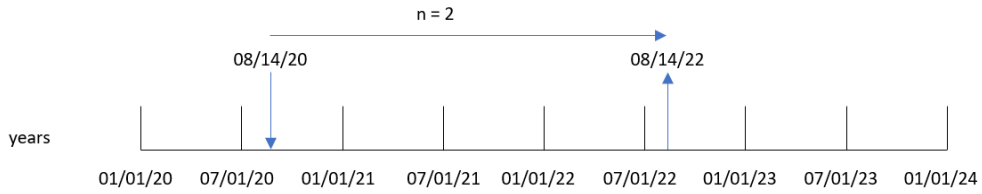
- date
- two_years_later

결과 테이블

date	two_years_later
01/10/2020	01/10/2022
02/28/2020	02/28/2022
04/09/2020	04/09/2022
04/16/2020	04/16/2022
05/21/2020	05/21/2022
08/14/2020	08/14/2022
10/07/2020	10/07/2022
12/05/2020	12/05/2022
01/22/2021	01/22/2023
02/03/2021	02/03/2023
03/17/2021	03/17/2023
04/23/2021	04/23/2023
05/04/2021	05/04/2023
06/30/2021	06/30/2023
07/26/2021	07/26/2023
12/27/2021	12/27/2023
02/02/2022	02/02/2024
02/26/2022	02/26/2024
03/07/2022	03/07/2024
03/11/2022	03/11/2024

'two_years_later' 필드는 addyears() 함수를 사용하여 선행 LOAD 문에서 만들어집니다. 제공된 첫 번째 인수는 평가 중인 날짜를 식별합니다. 두 번째 인수는 시작 날짜에서 더하거나 뺄 연도 수입니다. 이 경우 값 2가 제공됩니다.

addyears() 함수의 다이어그램, 기본 예



트랜잭션 8193은 2020년 8월 14일에 발생했습니다. 따라서 addyears() 함수는 two_years_later 필드에 대해 2022년 8월 14일을 반환합니다.

예 2 – 차트 개체 예

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Transactions라는 테이블에 로드되는 2020년과 2022년 사이의 트랜잭션 집합이 포함된 데이터 집합.
- DateFormat 시스템 변수(MM/DD/YYYY) 서식으로 제공된 날짜 필드.

차트 개체에서 트랜잭션이 발생한 1년 전 날짜를 반환하는 측정값 prior_year_date를 만듭니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다. date.

다음 측정값을 만들어 각 트랜잭션 1년 전의 날짜를 계산합니다.

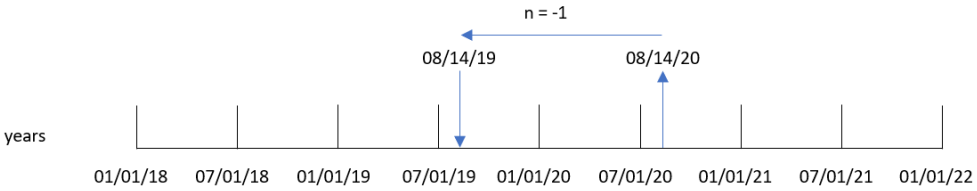
```
=addyears(date,-1)
```

결과 테이블

date	=addyears(date,-1)
01/10/2020	01/10/2019
02/28/2020	02/28/2019
04/09/2020	04/09/2019
04/16/2020	04/16/2019
05/21/2020	05/21/2019
08/14/2020	08/14/2019
10/07/2020	10/07/2019
12/05/2020	12/05/2019
01/22/2021	01/22/2020
02/03/2021	02/03/2020
03/17/2021	03/17/2020
04/23/2021	04/23/2020
05/04/2021	05/04/2020
06/30/2021	06/30/2020
07/26/2021	07/26/2020
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

one_year_prior 측정값은 addyears() 함수를 사용하여 차트 개체에 만들어집니다. 제공된 첫 번째 인수는 평가 중인 날짜를 식별합니다. 두 번째 인수는 startdate에서 더하거나 뺄 연도 수입니다. 이 경우 값 1가 제공됩니다.

addyears() 함수의 다이어그램, 차트 개체 예



트랜잭션 8193은 8월 14일에 발생했습니다. 따라서 addyears() 함수는 one_year_prior 필드에 대해 2019년 8월 14일을 반환합니다.

예 3 – 시나리오

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Warranties라는 테이블에 로드되는 데이터 집합.
- 제품 ID, 구매 날짜, 보증 기간 및 구매 가격 정보.

최종 사용자는 각 제품의 보증 종료 날짜를 제품 ID별로 표시하는 차트 개체를 원합니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- product_id
- purchase_date
- warranty_length

다음 측정값을 만들어 각 제품의 보증 종료 날짜를 계산합니다.

```
=addyears(purchase_date,warranty_length)
```

결과 테이블

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

product_id	purchase_date	warranty_length	=addyears(purchase_date,warranty_length)
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

age 함수는 **date_of_birth**에 태어난 사람이 **timestamp**의 시간에 (만으로) 몇 살인지 반환합니다.

구문:

```
age(timestamp, date_of_birth)
```

표현식일 수 있습니다.

반환 데이터 유형: 숫자

인수:

인수

인수	설명
timestamp	완전한 연도 수를 계산하는 데 사용되는 타임스탬프 또는 타임스탬프로 변환되는 표현식입니다.
date_of_birth	나이를 계산 중인 사람의 생년월일입니다. 표현식일 수 있습니다.

예 및 결과:
이 예에서는 날짜 서식 **DD/MM/YYYY**를 사용합니다. 날짜 서식은 데이터 로드 스크립트 맨 위에서 **SET DateFormat** 문으로 지정됩니다. 이 예제의 서식을 필요에 따라 변경하십시오.

스크립팅 예

예	결과
age('25/01/2014', '29/10/2012')	1을 반환합니다.
age('29/10/2014', '29/10/2012')	2을 반환합니다.

예제 스크립트를 앱에 추가하고 실행합니다. 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.

```
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;
```

결과 테이블에 테이블 내 각 레코드에 대한 age의 반환된 값이 표시됩니다.

결과 테이블

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

UTC 또는 GMT 타임스탬프를 이중 값 형태의 현지 시간으로 변환합니다. place는 전 세계 여러 도시, 장소, 표준 시간대로 지정할 수 있습니다.

구문:

```
ConvertToLocalTime(timestamp [, place [, ignore_dst=false]])
```

반환 데이터 유형: dual

인수

인수	설명
timestamp	변환할 타임스탬프 또는 타임스탬프로 평가되는 표현식입니다.
place	아래의 유효한 장소 및 표준 시간대 테이블에 있는 장소 또는 표준 시간대입니다. 또는 GMT나 UTC를 사용하여 현지 시간을 정의할 수 있습니다. 다음 값 및 시간 오프셋 범위를 사용할 수 있습니다. • 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 *DST 오프셋을 사용하는 경우(즉, False로 평가되는 **ignore_dst** 인수 값 지정) **place** 인수에 GMT 오프셋이 아닌 위치를 지정해야 합니다. 이는 일광 절약 시간을 조정하려면 GMT 오프셋에서 제공하는 경도 정보와 함께 위도 정보가 필요하기 때문입니다. 자세한 내용은 DST와 함께 GMT 오프셋 사용 (page 594)을 참조하십시오.* *표준 시간 오프셋만 사용할 수 있으며 GMT-04:27과 같은 임의의 시간 오프셋은 사용할 수 없습니다.*
ignore_dst	이 인수가 True로 평가되면 DST(일광 절약 시간)가 무시됩니다. True로 평가되는 유효한 인수 값에는 -1 및 True()이 포함됩니다. 이 인수가 False로 평가되면 일광 절약 시간에 맞게 타임스탬프가 조정됩니다. False로 평가되는 유효한 인수 값에는 0 및 False()이 포함됩니다. **ignore_dst** 인수 값이 잘못된 경우 함수는 **ignore_dst** 값이 True로 평가되는 것처럼 표현식을 평가합니다. **ignore_dst** 인수 값이 지정되지 않은 경우 함수는 **ignore_dst** 값이 False로 평가되는 것처럼 표현식을 평가합니다.

유효한 장소 및 표준 시간대

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	-	-	-

예 및 결과:

스크립팅 예

예	결과
`ConvertToLocalTime('2023-08-14 08:39:47','Paris')`	'2023-08-14 10:39:47' 및 해당하는 내부 타임스탬프 표현을 반환합니다.
`ConvertToLocalTime(UTC(), 'Stockholm')`	일광 절약 시간에 맞게 조정하여 스톡홀름의 시간을 반환합니다.
`ConvertToLocalTime(UTC(), 'Stockholm', -1)`	일광 절약 시간 조정 없이 스톡홀름의 시간을 반환합니다.
`ConvertToLocalTime(UTC(), 'GMT-05:00')`	북미 동부 해안(예: 뉴욕)의 시간을 반환합니다. 장소가 아닌 GMT 오프셋이 지정되기 때문에 일광 절약 시간에 대한 조정이 이루어지지 않습니다.
`ConvertToLocalTime(UTC(), 'New York', -1)`	일광 절약 시간 조정 없이 북미 동부 해안(뉴욕)의 시간을 반환합니다.
`ConvertToLocalTime(UTC(), 'New York', True())`	일광 절약 시간 조정 없이 북미 동부 해안(뉴욕)의 시간을 반환합니다.
`ConvertToLocalTime(UTC(), 'New York', 0)`	일광 절약 시간에 맞게 조정하여 북미 동부 해안(뉴욕)의 시간을 반환합니다.
`ConvertToLocalTime(UTC(), 'New York', False())`	일광 절약 시간에 맞게 조정하여 북미 동부 해안(뉴욕)의 시간을 반환합니다.

DST와 함께 GMT 오프셋 사용

Qlik Sense에서 ICU(International Components for Unicode) 라이브러리를 구현한 후 DST(일광 절약 시간)와 함께 GMT(그리니치 표준시) 오프셋을 사용하려면 추가 위도 정보가 필요합니다.

GMT는 경도(동-서) 오프셋인이며, DST는 위도(북-남) 오프셋입니다. 예를 들어, 헬싱키(핀란드)와 요하네스버그(남아프리카)는 동일한 GMT+02:00 오프셋을 공유하지만 동일한 DST 오프셋을 공유하지 않습니다. 즉, GMT 오프셋 외에도 모든 DST 오프셋에는 로컬 DST 조건에 대한 전체 정보를 얻기 위해 로컬 표준 시간대의 위도 위치(지리적 표준 시간대 입력)에 대한 정보가 필요합니다.

day

이 함수는 **expression**의 분위수가 표준 숫자 해석에 따라 날짜로 해석될 경우 일을 나타내는 정수를 반환합니다.

이 함수는 특정 날짜에 대한 해당 월의 날짜를 반환합니다. 일반적으로 캘린더 차원의 일부로 날짜 필드를 파생시키는 데 사용됩니다.

구문:

```
day(expression)
```

반환 데이터 유형: 정수

함수 예

예	결과
day(1971-10-12)	12를 반환합니다.
day(35648)	35648 = 1997-08-06이므로, 6을 반환합니다.

예 1 – DateFormat 데이터 집합(스크립트)

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Master_Calendar라는 날짜의 데이터 집합. DateFormat 시스템 변수는 DD/MM/YYYY로 설정됩니다.
- day_of_month 함수를 사용하여 day()라는 추가 필드를 만드는 선행 LOAD입니다.
- 전체 월 이름을 표현하는 date() 함수를 사용하는 long_date라는 추가 필드입니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- long_date
- day_of_month

결과 테이블

날짜	long_date	day_of_month
03/11/2022	11-March- 2022	11
03/12/2022	12-March- 2022	12
03/13/2022	13-March- 2022	13
03/14/2022	14-March- 2022	14
03/15/2022	15-March- 2022	15
03/16/2022	16-March- 2022	16
03/17/2022	17-March- 2022	17
03/18/2022	18-March- 2022	18
03/19/2022	19-March- 2022	19
03/20/2022	20-March- 2022	20
03/21/2022	21-March- 2022	21

날짜는 스크립트의 day() 함수에 의해 올바르게 평가됩니다.

예 2 – ANSI 날짜(스크립트)

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Master_Calendar라는 날짜의 데이터 집합. DateFormat 시스템 변수 DD/MM/YYYY가 사용됩니다. 그러나 데이터 집합에 포함된 날짜는 ANSI 표준 날짜 서식입니다.
- date() 함수를 사용하여 day_of_month라는 추가 필드를 만드는 선행 LOAD입니다.
- 전체 월 이름으로 날짜를 표현하는 date() 함수를 사용하는 long_date라는 추가 필드입니다.

로드 스크립트

```
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
 ];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- long_date
- day_of_month

결과 테이블

날짜	long_date	day_of_month
03/11/2022	11-March- 2022	11
03/12/2022	12-March- 2022	12
03/13/2022	13-March- 2022	13
03/14/2022	14-March- 2022	14
03/15/2022	15-March- 2022	15
03/16/2022	16-March- 2022	16

날짜	long_date	day_of_month
03/17/2022	17-March- 2022	17
03/18/2022	18-March- 2022	18
03/19/2022	19-March- 2022	19
03/20/2022	20-March- 2022	20
03/21/2022	21-March- 2022	21

날짜는 스크립트의 day() 함수에 의해 올바르게 평가됩니다.

예 3 – 형식이 지정되지 않은 날짜(스크립트)

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Master_Calendar라는 날짜의 데이터 집합. DateFormat 시스템 변수 DD/MM/YYYY가 사용됩니다.
- day() 함수를 사용하여 day_of_month라는 추가 필드를 만드는 선행 LOAD입니다.
- unformatted_date라는 형식이 지정되지 않은 원래 날짜입니다.
- date()를 사용하는 추가 필드 long_date는 숫자형 날짜를 형식이 지정된 날짜 필드로 변환하는 데 사용됩니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- unformatted_date
- long_date
- day_of_month

결과 테이블

unformatted_date	long_date	day_of_month
44868	03-November- 2022	3
44898	03-December- 2022	3
44928	02-January- 2023	2
44958	01-February- 2023	1
44988	03-March- 2023	3
45008	23-March- 2023	23
45018	02-April- 2023	2
45038	22-April- 2023	22
45048	02-May- 2023	2
45068	22-May- 2023	22
45078	01-June- 2023	1

날짜는 스크립트의 day() 함수에 의해 올바르게 평가됩니다.

예 4 - 만료 월 계산(차트)

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 3월에 이루어진 주문 데이터 집합(Orders라고 함). 테이블에는 세 개의 필드가 있습니다.
  - id
  - order_date
  - amount

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다.order_date.

배송 날짜를 계산하려면 다음 측정값 =day(order_date+5)를 만듭니다.

결과 테이블

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

day() 함수는 3월 11일의 주문이 5일 배송 기간을 기준으로 16일에 배송될 것이라고 올바르게 결정합니다.

dayend

이 함수는 **time**에 포함된 날의 마지막 밀리초의 타임스탬프에 해당하는 값을 반환합니다. 기본 출력 형식은 스크립트에 설정된 **TimestampFormat**입니다.

구문:

```
DayEnd(time[, [period_no[, day_start]])
```

사용 시기

dayend() 함수는 일반적으로 사용자가 아직 발생하지 않은 하루 중 시간에 대한 분수를 사용하여 계산하려고 할 때 표현식의 일부로 사용됩니다. 예를 들어, 하루 동안 발생되는 총 비용을 계산합니다.

반환 데이터 유형: dual

인수

인수	설명
time	평가할 타임스탬프입니다.
period_no	**period_no**는 정수 또는 정수로 처리되는 표현식이며, 값 0은 **time**을 포함하는 날을 나타냅니다. **period_no**가 음수 값일 경우 이전 날, 양수 값일 경우 이후 날을 나타냅니다.
day_start	하루가 자정에 시작되지 않도록 지정하려면 **day_start**에 하루의 분위수로 오프셋을 지정합니다. 예를 들어 0.125는 오전 3:00을 나타냅니다. 즉, 오프셋을 만들려면 시작 시간을 24시간으로 나눕니다. 예를 들어, 하루를 오전 7시에 시작하도록 하려면 분수 7/24를 사용합니다.

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

함수 예

예	결과
dayend('2013/01/25 16:45:00')	2013/01/25 23:59:59를 반환합니다. PM
dayend('2013/01/25 16:45:00', -1)	2013/01/24 23:59:59를 반환합니다. PM
dayend('2013/01/25 16:45:00', 0, 0.5)	2013/01/26 11:59:59를 반환합니다. PM

예 1 - 기본 스크립트

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 날짜 목록이 포함된 데이터 집합이 "Calendar"라는 테이블에 로드됩니다.
- 기본 DateFormat 시스템 변수 (MM/DD/YYYY).
- dayend() 함수를 사용하여 추가 필드 'EOD_timestamp'를 만들기 위한 선행 LOAD입니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- EOD_timestamp

결과 테이블

날짜	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
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

위의 표에서 볼 수 있듯이 데이터 집합의 각 날짜에 대해 하루의 종료 타임스탬프가 생성됩니다. 타임스탬프는 시스템 변수 형식 `TimestampFormat M/D/YYYY h:mm:ss[.fff] TT`입니다.

예 2 – period_no

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

서비스 예약이 포함된 데이터 집합을 'Services'라는 테이블에 로드합니다.

데이터 집합에는 다음 필드가 포함됩니다.

- `service_id`
- `service_date`
- `amount`

테이블에 두 개의 새 필드를 만듭니다.

- `deposit_due_date`: 보증금을 받아야 하는 날짜입니다. 이는 `service_date` 3일 전 날짜의 끝입니다.
- `final_payment_due_date`: 최종 지급을 받아야 하는 날짜입니다. 이는 `service_date` 7일 후 날짜의 끝입니다.

위의 두 필드는 `dayend()` 함수를 사용하여 이전 로드에서 만들어지고 처음 두 매개 변수 `time` 및 `period_no`를 제공합니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- service_date
- deposit_due_date
- final_payment_due_date

결과 테이블

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

service_date	deposit_due_date	final_payment_due_date
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

새 필드의 값은 TimestampFormat M/D/YYYY h:mm:ss[.fff] TT에 있습니다. 함수 dayend()가 사용되었으므로 타임스탬프 값은 모두 그날의 마지막 밀리초입니다.

dayend() 함수에 전달된 두 번째 인수가 음수이므로 보증금 기한 값 값은 서비스 날짜 3일 전입니다.

dayend() 함수에 전달된 두 번째 인수가 양수이므로 최종 지급 기한 값은 서비스 날짜 7일 후입니다.

예 3 – day_start 스크립트

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

이 예에서 사용된 데이터 집합 및 시나리오는 이전 예와 동일합니다.

이전 예에서와 같이 두 개의 새 필드를 만듭니다.

- deposit_due_date: 보증금을 받아야 하는 날짜입니다. 이는 service_date 3일 전 날짜의 끝입니다.
- final_payment_due_date: 최종 지급을 받아야 하는 날짜입니다. 이는 service_date 7일 후 날짜의 끝입니다.

그러나 회사에서 근무일이 오후 5시에 시작하여 다음 날 오후 5시에 끝나는 정책으로 운영하려고 합니다. 그러면 회사에서 해당 근무 시간에 발생하는 트랜잭션을 모니터링할 수 있습니다.

이러한 요구 사항을 달성하기 위해 위의 두 필드는 dayend() 함수를 사용하여 이전 로드에서 만들어지고 세 인수 time, period_no 및 day_start를 모두 사용합니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- service_date
- deposit_due_date
- final_payment_due_date

결과 테이블

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

날짜는 예 2와 동일하게 유지되지만 dayend() 함수에 전달된 세 번째 인수 day_start의 값이 17/24였으므로 날짜에는 이제 오후 5시 이전의 마지막 밀리초의 타임스탬프가 있습니다.

예 4 – 차트 예

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

이 예에 사용된 데이터 집합과 시나리오는 앞의 두 예와 동일합니다. 회사에서 근무일이 오후 5시에 시작하여 다음 날 오후 5시에 끝나는 정책으로 운영하려고 합니다.

이전 예에서와 같이 두 개의 새 필드를 만듭니다.

- deposit_due_date: 보증금을 받아야 하는 날짜입니다. 이는 service_date 3일 전 날짜의 끝입니다.
- final_payment_due_date: 최종 지급을 받아야 하는 날짜입니다. 이는 service_date 7일 후 날짜의 끝입니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다.

service_date.

deposit_due_date 필드를 만들려면 다음 측정값을 만듭니다.

```
=dayend(service_date,-3,17/24).
```

그런 다음 final_payment_due_date 필드를 만들려면 다음 측정값을 만듭니다.

```
=dayend(service_date,7,17/24).
```

결과 테이블

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

새 필드의 값은 `TimestampFormat M/D/YYYY h:mm:ss[.fff] TT`에 있습니다. 함수 `dayend()`가 사용되었으므로 타임스탬프 값은 모두 그날의 마지막 밀리초입니다.

`dayend()` 함수에 전달된 두 번째 인수가 음수이므로 지급 기한 값은 서비스 날짜 3일 전입니다.

`dayend()` 함수에 전달된 두 번째 인수가 양수이므로 최종 지급 기한 값은 서비스 날짜 7일 후입니다.

`dayend()` 함수에 전달된 세 번째 인수 `day_start`의 값이 17/24였으므로 날짜에는 오후 5시 이전의 마지막 밀리초의 타임스탬프가 있습니다.

인수

인수	설명
time	평가할 타임스탬프입니다.
period_no	**period_no**는 정수 또는 정수로 처리되는 표현식이며, 값 0은 **time**을 포함하는 날을 나타냅니다. **period_no**가 음수 값일 경우 이전 날, 양수 값일 경우 이후 날을 나타냅니다.
day_start	하루가 자정에 시작되지 않도록 지정하려면 **day_start**에 하루의 분위수로 오프셋을 지정합니다. 예를 들어 0.125는 오전 3:00을 나타냅니다.

daylightsaving

Windows에 정의된 일광 절약 시간제 시간의 현재 조정 내용을 반환합니다.

구문:

```
DaylightSaving( )
```

반환 데이터 유형: dual

```
daylightsaving( )
```

dayname

이 함수는 **time**을 포함한 날의 첫 번째 밀리초의 타임스탬프에 해당하는 기본 숫자 값으로 날짜를 표시하는 값을 반환합니다.

구문:

```
DayName(time[, period_no [, day_start]])
```

반환 데이터 유형: dual

인수:

인수

인수	설명
time	평가할 타임스탬프입니다.
period_no	**period_no**는 정수 또는 정수로 처리되는 표현식이며, 값 0은 **time**을 포함하는 날을 나타냅니다. **period_no**가 음수 값일 경우 이전 날, 양수 값일 경우 이후 날을 나타냅니다.
day_start	하루가 자정에 시작되지 않도록 지정하려면 **day_start**에 하루의 분위수로 오프셋을 지정합니다. 예를 들어 0.125는 오전 3:00을 나타냅니다.

예 및 결과:
이 예에서는 날짜 서식 **DD/MM/YYYY**를 사용합니다. 날짜 서식은 데이터 로드 스크립트 맨 위에서 **SET DateFormat** 문으로 지정됩니다. 이 예제의 서식을 필요에 따라 변경하십시오.

스크립팅 예

예	결과
`dayname('25/01/2013 16:45:00')`	25/01/2013를 반환합니다.
`dayname('25/01/2013 16:45:00', -1)`	24/01/2013를 반환합니다.
`dayname('25/01/2013 16:45:00', 0, 0.5 )`	25/01/2013을 반환합니다. 전체 타임스탬프를 표시하면 '25/01/2013 12:00:00.000.'에 해당하는 기본 숫자 값이 표시됩니다.

예제 스크립트를 앱에 추가하고 실행합니다. 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.

이 예에서는 테이블의 각 송장 날짜 다음 날짜의 시작을 나타내는 타임스탬프에서 날짜 이름을 만듭니다.

```
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;
```

결과 테이블에는 원래 날짜와 dayname() 함수의 반환 값이 표시된 열이 포함됩니다. 속성 패널에서 서식을 지정하면 전체 타임스탬프를 표시할 수 있습니다.

결과 테이블

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

이 함수는 타임스탬프가 속하는 분기의 일수를 계산합니다. 마스터 캘린더를 만들 때 사용하는 함수입니다.

구문:

```
DayNumberOfQuarter(timestamp[,start_month])
```

반환 데이터 유형: 정수

인수

인수	설명
timestamp	평가할 날짜 또는 타임스탬프입니다.
start_month	**start_month**를 2와 12 사이(생략할 경우 1)로 지정하면 연도의 시작일을 원하는 달의 첫 날로 옮길 수 있습니다. 예를 들어 3월 1일에 시작되는 회계년도를 사용하려는 경우 **start_month** = 3을 지정합니다.

이 예에서는 날짜 서식 **DD/MM/YYYY**를 사용합니다. 날짜 서식은 데이터 로드 스크립트 맨 위에서 **SET DateFormat** 문으로 지정됩니다. 이 예제의 서식을 필요에 따라 변경하십시오.

함수 예

예	결과
`DayNumberOfQuarter('12/09/2014')`	현재 분기의 일수인 74를 반환합니다.

예	결과
DayNumberOfQuarter ('12/09/2014',3)	현재 분기의 일수인 12를 반환합니다. 이 경우, 첫 번째 분기는 3월(start_month가 3으로 지정됨)부터 시작됩니다. 따라서 현재 분기는 9월 1일부터 시작된 3분기입니다.

예 1 – 1월 연도 시작(스크립트)

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Calendar라는 테이블에 로드되는 날짜 목록이 포함된 간단한 데이터 집합. 기본 DateFormat 시스템 변수 MM/DD/YYYY가 사용됩니다.
- DayNumberOfQuarter() 함수를 사용하여 DayNrQtr라는 추가 필드를 만드는 선행 LOAD입니다.

날짜를 제외하고 함수에 추가 매개 변수가 제공되지 않습니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- daynrqtr

결과 테이블

날짜	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

DayNumberOfQuarter() 함수에 전달된 두 번째 인수가 없으므로 해당 연도의 첫 번째 날은 1월 1일입니다.

1월 1일은 분기의 첫 번째 날이고 2월 1일은 분기의 32번째 날입니다. 3월 31일은 분기의 91번째 날이자 마지막 날이고 4월 1일은 2분기의 첫 번째 날입니다.

예 2 – 2월 연도 시작(스크립트)

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 첫 번째 예와 동일한 데이터 집합.
- 기본 DateFormat 시스템 변수 MM/DD/YYYY가 사용됩니다.
- 2월 1일에 시작하는 start_month 인수입니다. 이렇게 하면 회계 연도가 2월 1일로 설정됩니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- daynrqtr

결과 테이블

날짜	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

DayNumberOfQuarter() 함수에 전달된 두 번째 인수가 2였으므로 해당 연도의 첫 번째 날은 2월 1일입니다.

1분기는 2월에서 4월 사이에, 4분기는 11월에서 1월 사이에 운영됩니다. 이는 2월 1일이 분기의 첫 번째 날이고 1월 31일이 분기의 92번째 날이자 마지막 날로 결과 테이블에 표시됩니다.

예 3 – 1월 연도 시작(차트)

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 첫 번째 예와 동일한 데이터 집합.
- 기본 DateFormat 시스템 변수 MM/DD/YYYY가 사용됩니다.

그러나 이 예에서는 변경되지 않은 데이터 집합이 응용 프로그램에 로드됩니다. 분기의 날짜 값은 차트 개체의 측정값을 통해 계산됩니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다. date.

다음 측정값을 만듭니다.

```
=daynumberofquarter(date)
```

결과 테이블

날짜	=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

DayNumberOfQuarter() 함수에 전달된 두 번째 인수가 없으므로 해당 연도의 첫 번째 날은 1월 1일입니다.

1월 1일은 분기의 첫 번째 날이고 2월 1일은 분기의 32번째 날입니다. 3월 31일은 분기의 91번째 날이자 마지막 날이고 4월 1일은 2분기의 첫 번째 날입니다.

예 4 – 2월 연도 시작(차트)

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 첫 번째 예와 동일한 데이터 집합.
- 기본 DateFormat 시스템 변수 MM/DD/YYYY가 사용됩니다.
- 회계 연도는 2월 1일부터 1월 31일까지로 합니다.

그러나 이 예에서는 변경되지 않은 데이터 집합이 응용 프로그램에 로드됩니다. 분기의 날짜 값은 차트 개체의 측정값을 통해 계산됩니다.

로드 스크립트

```
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
];
```

차트 개체

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다. date.

다음 측정값을 만듭니다.

```
=daynumberofquarter(date,2)
```

결과

결과 테이블

날짜	=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

`DayNumberOfQuarter()` 함수에 전달된 두 번째 인수가 2였으므로 해당 연도의 첫 번째 날은 1월 1일입니다.

1분기는 2월에서 4월 사이에, 4분기는 11월에서 1월 사이에 운영됩니다. 이는 2월 1일이 분기의 첫 번째 날이고 1월 31일이 분기의 92번째 날이자 마지막 날로 결과 테이블에 표시됩니다.

daynumberofyear

이 함수는 타임스탬프가 속하는 연도의 일수를 계산합니다. 해당 연도의 첫 날의 첫 번째 밀리초로부터 계산되지만 첫 번째 월은 오프셋 지정할 수 있습니다.

구문:

```
DayNumberOfYear(timestamp[,start_month])
```

반환 데이터 유형: 정수

인수

인수	설명
timestamp	평가할 날짜 또는 타임스탬프입니다.
start_month	**start_month**를 2와 12 사이(생략할 경우 1)로 지정하면 연도의 시작일을 원하는 달의 첫 날로 옮길 수 있습니다. 예를 들어 3월 1일에 시작되는 회계년도를 사용하려는 경우 **start_month** = 3을 지정합니다.

이 예에서는 날짜 서식 **DD/MM/YYYY**를 사용합니다. 날짜 서식은 데이터 로드 스크립트 맨 위에서 **SET DateFormat** 문으로 지정됩니다. 이 예제의 서식을 필요에 따라 변경하십시오.

함수 예

예	결과
DayNumberOfYear('12/09/2014')	해당 연도의 1일부터 계수한 일수인 256을 반환합니다.
DayNumberOfYear('12/09/2014',3)	3월 1일부터 계수된 일수인 196을 반환합니다.

예 1 – 1월 연도 시작(스크립트)

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Calendar라는 테이블에 로드되는 날짜 목록이 포함된 간단한 데이터 집합. 기본 DateFormat 시스템 변수 MM/DD/YYYY가 사용됩니다.
- DayNumberOfYear() 함수를 사용하여 daynryear라는 추가 필드를 만드는 선행 LOAD입니다.

날짜를 제외하고 함수에 추가 매개 변수가 제공되지 않습니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- daynryear

결과 테이블

날짜	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

DayNumberOfYear() 함수에 전달된 두 번째 인수가 없으므로 해당 연도의 첫 번째 날은 1월 1일입니다.

1월 1일은 분기의 첫 번째 날이고 2월 1일은 해당 연도의 32번째 날입니다. 6월 30일은 182번째 날이고 12월 31일은 366번째 날이자 해당 연도의 마지막 날입니다.

예 2 – 11월 연도 시작(스크립트)

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 첫 번째 예와 동일한 데이터 집합.
- 기본 DateFormat 시스템 변수 MM/DD/YYYY가 사용됩니다.
- 11월 1일에 시작하는 start_month 인수입니다. 이렇게 하면 회계 연도가 11월 1일로 설정됩니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- daynryear

결과 테이블

날짜	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

DayNumberOfYear() 함수에 전달된 두 번째 인수가 11이었으므로 해당 연도의 첫 번째 날은 11월 1일입니다.

1월 1일은 분기의 첫 번째 날이고 2월 1일은 해당 연도의 32번째 날입니다. 6월 30일은 182번째 날이고 12월 31일은 366번째 날이자 해당 연도의 마지막 날입니다.

예 3 – 1월 연도 시작(차트)

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 첫 번째 예와 동일한 데이터 집합.
- 기본 DateFormat 시스템 변수 MM/DD/YYYY가 사용됩니다.

그러나 이 예에서는 변경되지 않은 데이터 집합이 응용 프로그램에 로드됩니다. 분기의 날짜 값은 차트 개체의 측정값을 통해 계산됩니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다. date.

다음 측정값을 만듭니다.

```
=daynumberofyear(date)
```

결과 테이블

날짜	=daynumberofyear(date)
01/01/2022	1
01/10/2022	10

날짜	=daynumberofyear(date)
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

DayNumberOfYear() 함수에 전달된 두 번째 인수가 없으므로 해당 연도의 첫 번째 날은 1월 1일입니다.

1월 1일은 해당 연도의 첫 번째 날이고 2월 1일은 해당 연도의 32번째 날입니다. 6월 30일은 182번째 날이고 12월 31일은 366번째 날이자 해당 연도의 마지막 날입니다.

예 4 – 11월 연도 시작(차트)

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 첫 번째 예와 동일한 데이터 집합.
- 기본 DateFormat 시스템 변수 MM/DD/YYYY가 사용됩니다.
- 회계 연도는 11월 1일부터 10월 31일까지로 합니다.

그러나 이 예에서는 변경되지 않은 데이터 집합이 응용 프로그램에 로드됩니다. 해당 연도의 날짜 값은 차트 개체의 측정값을 통해 계산됩니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다. date.

다음 측정값을 만듭니다.

```
=daynumberofyear(date)
```

결과 테이블

날짜	=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
10/31/2022	366
11/01/2022	1
12/31/2022	61

DayNumberOfYear() 함수에 전달된 두 번째 인수가 11이었으므로 해당 연도의 첫 번째 날은 11월 1일입니다.

회계 연도는 11월과 10월 사이에 운영됩니다. 이는 11월 1일이 해당 연도의 첫 번째 날이고 10월 31일이 366번째 날이자 해당 연도의 마지막 날로 결과 테이블에 표시됩니다.

daystart

이 함수는 **time** 인수에 포함된 날의 첫 번째 밀리초가 있는 타임스탬프에 해당하는 값을 반환합니다. 기본 출력 형식은 스크립트에 설정된 **TimestampFormat**입니다.

구문:

```
DayStart(time[, [period_no[, day_start]])
```

반환 데이터 유형: dual

인수

인수	설명
time	평가할 타임스탬프입니다.
period_no	**period_no**는 정수 또는 정수로 처리되는 표현식이며, 값 0은 **time**을 포함하는 날을 나타냅니다. **period_no**가 음수 값일 경우 이전 날, 양수 값일 경우 이후 날을 나타냅니다.
day_start	하루가 자정에 시작되지 않도록 지정하려면 **day_start**에 하루의 분위수로 오프셋을 지정합니다. 예를 들어 0.125는 오전 3:00을 나타냅니다. 즉, 오프셋을 만들려면 시작 시간을 24시간으로 나눕니다. 예를 들어, 하루를 오전 7시에 시작하도록 하려면 분수 7/24를 사용합니다.

사용 시기

daystart() 함수는 일반적으로 사용자가 지금까지 경과된 날짜의 분수를 사용하여 계산하려고 할 때 표현식의 일부로 사용됩니다. 예를 들어, 지금까지 하루에 직원이 받은 총 급여를 계산하는 데 사용할 수 있습니다.

이 예에서는 타임스탬프 형식 'M/D/YYYY h:mm:ss[.fff] TT'을 사용합니다. 타임스탬프 형식은 데이터 로드 스크립트 맨 위에서 SET TimeStamp 문으로 지정됩니다. 이 예의 형식을 필요에 따라 변경하십시오.

함수 예

예	결과
daystart('01/25/2013 4:45:00 PM')	1/25/2013 12:00:00 AM를 반환합니다.
daystart('1/25/2013 4:45:00 PM', -1)	1/24/2013 12:00:00 AM를 반환합니다.
daystart('1/25/2013 16:45:00',0,0.5)	1/25/2013 12:00:00 PM를 반환합니다.

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

예 1 – 간단한 예

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Calendar라는 테이블에 로드되는 날짜 목록이 포함된 간단한 데이터 집합.
- 기본 TimeStampFormat 시스템 변수(M/D/YYYY h:mm:ss[.fff] TT)가 사용됩니다.
- daystart() 함수를 사용하여 SOD_timestamp라는 추가 필드를 만드는 선행 LOAD입니다.

날짜를 제외하고 함수에 추가 매개 변수가 제공되지 않습니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- SOD_timestamp

결과 테이블

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

위의 표에서 볼 수 있듯이 데이터 집합의 각 날짜에 대해 하루의 종료 타임스탬프가 생성됩니다. 타임스탬프는 시스템 변수 형식 `TimestampFormat M/D/YYYY h:mm:ss[.fff] TT`입니다.

예 2 – period_no

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- `Fines`라는 테이블에 로드되는 주차 벌금이 포함된 데이터 집합. 데이터 집합에는 다음 필드가 포함됩니다.
  - `id`
  - `due_date`
  - `number_plate`
  - `amount`
- `daystart()` 함수를 사용하고 세 가지 매개 변수(`time`, `period_no` 및 `day_start`)를 모두 제공하는 선행 LOAD. 이 선행 LOAD는 다음과 같은 두 개의 새 날짜 필드를 만듭니다.
  - 지급 기한 7일 전부터 시작되는 `early_repayment_period` 날짜 필드.
  - 지급 기한 14일 후에 시작되는 `late_penalty_period` 날짜 필드.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- due_date
- early_repayment_period
- late_penalty_period

결과 테이블

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

새 필드의 값은 TimestampFormat M/DD/YYYY tt에 있습니다. 함수 daystart()가 사용되었으므로 타임스탬프 값은 모두 그날의 첫 번째 밀리초입니다.

조기 상환 기간 값은 daystart() 함수에 전달된 두 번째 인수가 음수이므로 기한 7일 전입니다.

지연 상환 기간 값은 daystart() 함수에 전달된 두 번째 인수가 양수이므로 기한 14일 후입니다.

예 3 – day_start

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 이전 예와 동일한 데이터 집합 및 시나리오입니다.
- 이전 예와 동일한 선행 LOAD입니다.

이 예에서는 매일 오전 7시에 시작하고 종료하도록 근무일을 설정합니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- due_date
- early_repayment_period
- late_penalty_period

결과 테이블

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

daystart() 함수에 전달된 day_start 인수의 값이 7/24였으므로 날짜에는 오전 7:00의 타임스탬프가 있습니다. 이렇게 하면 날짜의 시작이 오전 7:00로 설정됩니다.

due_date 필드에 타임스탬프가 없기 때문에 오전 12:00로 처리됩니다. 이는 날짜가 오전 7:00에 시작하고 종료되므로 여전히 전날의 일부입니다. 따라서 2월 11일 벌금에 대한 조기 상환 기간은 2월 3일 오전 7:00부터 시작됩니다.

예 4 - 차트 개체 예

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

이 예에서는 이전 예와 동일한 데이터 집합 및 시나리오를 사용합니다.

그러나 원본 Fines 테이블만 응용 프로그램에 로드되고 두 개의 추가 기한 값이 차트 개체에서 계산됩니다.

로드 스크립트

```
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
];
```

결과

다음과 같이 하십시오.

1. 데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다. due_date.
2. early_repayment_period 필드를 만들려면 다음 측정값을 만듭니다.
   =daystart(due_date,-7,7/24)
3. late_penalty_period 필드를 만들려면 다음 측정값을 만듭니다.
   =daystart(due_date,14,7/24)

결과 테이블

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

새 필드의 값은 TimestampFormat M/D/YYYY h:mm:ss[.fff] TT에 있습니다. daystart() 함수가 사용되었으므로 타임스탬프 값은 해당 날짜의 첫 번째 밀리초에 해당합니다.

조기 상환 기간 값은 daystart() 함수에 전달된 두 번째 인수가 음수였으므로 기한 7일 전입니다.

지연 상환 기간 값은 daystart() 함수에 전달된 두 번째 인수가 양수였으므로 기한 14일 후입니다.

daystart() 함수에 전달된 세 번째 인수의 값 day_start가 7/24였으므로 날짜에는 오전 7:00의 타임스탬프가 있습니다.

firstworkdate

firstworkdate 함수는 선택적으로 나열된 공휴일을 고려하여 **no_of_workdays**(월요일 ~ 금요일)가 **end_date** 이전에 끝나게 되는 가장 최근의 시작 날짜를 반환합니다. **end_date** 및 **holiday**은 유효한 날짜 또는 타임스태프여야 합니다.

구문:

```
firstworkdate(end_date, no_of_workdays {, holiday} )
```

반환 데이터 유형: 정수

인수:

인수

인수	설명
end_date	평가할 끝 날짜의 타임스탬프입니다.
no_of_ workdays	근무해야 할 일수입니다.
holiday	근무일에서 제외시킬 공휴일 기간입니다. 휴일은 문자열 상수 날짜로 표시됩니다. 여러 휴일 날짜를 쉼표로 구분하여 지정할 수 있습니다. **'12/25/2013', '12/26/2013', '12/31/2013', '01/01/2014'**

예 및 결과:
이 예에서는 날짜 서식 **DD/MM/YYYY**를 사용합니다. 날짜 서식은 데이터 로드 스크립트 맨 위에서 **SET DateFormat** 문으로 지정됩니다. 이 예제의 서식을 필요에 따라 변경하십시오.

스크립팅 예

예	결과
`firstworkdate ('29/12/2014', 9)`	'17/12/2014를 반환합니다.
`firstworkdate ('29/12/2014', 9, '25/12/2014', '26/12/2014')`	2일 간의 휴일을 고려하여 15/12/2014를 반환합니다.

예제 스크립트를 앱에 추가하고 실행합니다. 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.

```
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;
```

결과 테이블에 테이블 내 각 레코드에 대한 FirstWorkDate의 반환된 값이 표시됩니다.

결과 테이블

InvID	EndDate	StartDate
1	28/03/2015	13/10/2014
2	10/12/2015	26/06/2015
3	5/2/2016	24/08/2015
4	31/3/2016	16/10/2015
5	19/5/2016	04/12/2015
6	15/9/2016	01/04/2016

GMT

이 함수는 지역 설정에서 유추한 현재 Greenwich Mean Time을 반환합니다. 이 함수는 TimestampFormat 시스템 변수 서식으로 값을 반환합니다.

앱이 다시 로드될 때마다 GMT 함수를 사용하는 모든 로드 스크립트 테이블, 변수 또는 차트 개체는 시스템 시계에서 파생된 최신 현재 그리니치 표준시로 조정됩니다.

구문:

```
GMT( )
```

반환 데이터 유형: dual

이 예에서는 타임스탬프 형식 M/D/YYYY h:mm:ss[.fff] TT을 사용합니다. 날짜 서식은 데이터 로드 스크립트 맨 위에서 SET TimestampFormat 문으로 지정됩니다. 이 예의 형식을 필요에 따라 변경하십시오.

함수 예

예	결과
GMT()	3/28/2022 2:47:36 PM

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

예 1 - 변수(스크립트)

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다. 이 예에서는 GMT 함수를 사용하여 로드 스크립트에서 현재 그리니치 표준시를 변수로 설정합니다.

로드 스크립트

```
LET vGMT = GMT();
```

결과

데이터를 로드하고 시트를 만듭니다. **텍스트 및 이미지** 차트 개체를 사용하여 텍스트 상자를 만듭니다.

이 측정값을 텍스트 상자에 추가합니다.

```
=vGMT
```

텍스트 상자에는 아래와 같이 날짜와 시간이 포함된 텍스트 줄이 포함되어야 합니다.

```
3/28/2022 2:47:36 PM
```

예 2 - 11월 연도 시작(스크립트)

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Overdue라는 테이블에 로드되는 연체된 도서관 도서가 포함된 데이터 집합. 기본 DateFormat 시스템 변수 MM/DD/YYYY가 사용됩니다.
- 각 도서의 연체일을 계산하는 days_overdue라는 새 필드 만들기.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- due_date
- book_id
- days_overdue

결과 테이블

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

days_overdue 필드의 값은 GMT() 함수를 사용하여 현재 그리니치 표준시와 원본 기한 간의 차이를 찾아 계산합니다. 날짜만 계산하기 위해 결과는 Floor() 함수를 사용하여 가장 가까운 정수로 반올림됩니다.

예 3 - 차트 개체(차트)

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다. 로드 스크립트에는 이전 예와 동일한 데이터 집합이 포함되어 있습니다. 기본 DateFormat 시스템 변수 MM/DD/YYYY가 사용됩니다.

그러나 이 예에서는 변경되지 않은 데이터 집합이 응용 프로그램에 로드됩니다. 연체 일수 값은 차트 개체의 측정값을 통해 계산됩니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- due_date
- book_id

다음 측정값을 만듭니다.

```
=Floor(GMT() - due_date)
```

결과 테이블

due_date	book_id	=Floor(GMT()-due_date)
01/01/2021	4	455
01/10/2021	24	446

due_date	book_id	=Floor(GMT()-due_date)
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

days_overdue 필드의 값은 GMT() 함수를 사용하여 현재 그리니치 표준시와 원본 기한 간의 차이를 찾아 계산합니다. 날짜만 계산하기 위해 결과는 Floor() 함수를 사용하여 가장 가까운 정수로 반올림됩니다.

hour

이 함수는 **expression**의 분위수가 표준 숫자 해석에 따라 시간으로 해석될 경우 시간을 나타내는 정수를 반환합니다.

구문:

```
hour(expression)
```

반환 데이터 유형: 정수

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

함수 예

예	결과
hour('09:14:36')	제공된 텍스트 문자열은 TimestampFormat 변수에 정의된 타임스탬프 형식과 일치하므로 암시적으로 타임스탬프로 변환됩니다. 표현식은 9를 반환합니다.
hour('0.5555')	표현식은 13을 반환합니다(0.5555 = 13:19:55).

예 1 – 변수(스크립트)

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 타임스탬프별 트랜잭션이 포함된 데이터 집합.
- 기본 TimeStamp 시스템 변수 (M/D/YYYY h:mm:ss[.fff] TT)

구매가 발생하는 시점을 계산하는 'hour' 필드를 만듭니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- hour

결과 테이블

날짜	hour
2022-01-01 22:10:22	22
2022-01-02 08:35:54	8
2022-01-03 05:40:49	5
2022-01-03 14:21:53	14
2022-01-04 18:49:38	18
2022-01-04 22:58:34	22
2022-01-05 19:04:57	19
2022-01-05 19:34:46	19
2022-01-06 08:49:09	8
2022-01-06 11:29:38	11

시간 필드의 값은 hour() 함수를 사용하고 선행 LOAD 문의 표현식으로 날짜를 전달하여 만들어집니다.

예 2 – 차트 개체(차트)

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 첫 번째 예와 동일한 데이터 집합.
- 기본 TimeStamp 시스템 변수 (M/D/YYYY h:mm:ss[.fff] TT).

그러나 이 예에서는 변경되지 않은 데이터 집합이 응용 프로그램에 로드됩니다. 'hour' 값은 차트 개체의 측정값을 통해 계산됩니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다. date.

'hour'를 계산하려면 다음 측정값을 만듭니다.

```
=hour(date)
```

결과 테이블

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

'hour'의 값은 hour() 함수를 사용하고 차트 개체에 대한 측정값의 표현식으로 날짜를 전달하여 만들어집니다.

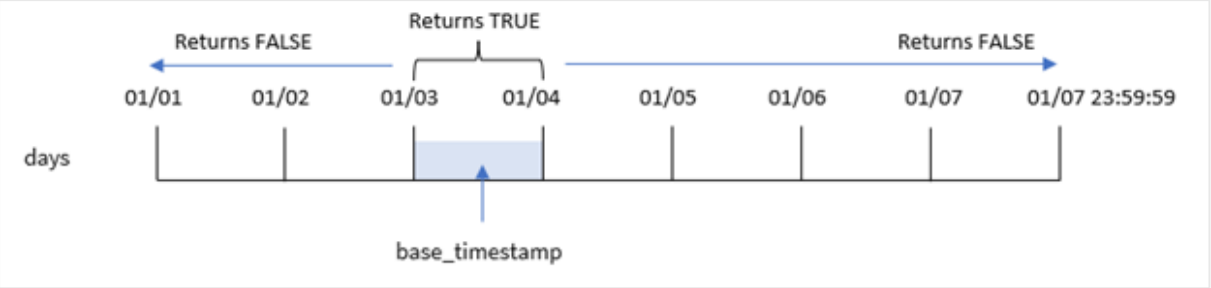
inday

timestamp가 **base_timestamp**를 포함하는 날에 속할 경우 이 함수는 True를 반환합니다.

구문:

```
InDay (timestamp, base_timestamp, period_no[, day_start])
```

inday 함수 다이어그램



inday() 함수는 base_timestamp 인수를 사용하여 타임스탬프가 속하는 날짜를 식별합니다. 날짜의 시작 시간은 기본적으로 자정입니다. 그러나 inday() 함수의 day_start 인수를 사용하여 날짜의 시작 시간을 변경할 수 있습니다. 이 날짜가 정의되면 함수는 규정된 타임스탬프 값을 해당 날짜와 비교할 때 부울 결과를 반환합니다.

사용 시기

inday() 함수는 부울 결과를 반환합니다. 일반적으로 이 유형의 함수는 if expression의 조건으로 사용됩니다. 평가된 날짜가 해당 타임스탬프의 날짜에 발생했는지 여부에 따라 집계 또는 계산을 반환합니다.

예를 들어, inday() 함수를 사용하여 지정된 날짜에 제조된 모든 장비를 식별할 수 있습니다.

반환 데이터 유형: 부울

Qlik Sense에서 부울 true 값은 -1로 표시되고 false 값은 0으로 표시됩니다.

인수

인수	설명
timestamp	base_timestamp와 비교할 날짜 및 시간입니다.
base_timestamp	타임스탬프를 평가하는 데 사용되는 날짜와 시간입니다.
period_no	일은 period_no로 오프셋을 지정할 수 있습니다. period_no는 정수이며, 값 0은 base_timestamp를 포함하는 날을 나타냅니다. period_no가 음수 값일 경우 이전 날, 양수 값일 경우 이후 날을 나타냅니다.
day_start	작업일이 자정부터 시작되지 않도록 하려면 day_start에 일의 분위수 형식으로 오프셋을 지정하십시오(예: 0.125 = 오전 3시).

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

함수 예

예	결과
inday ('01/12/2006 12:23:00 PM', '01/12/2006 12:00:00 AM', 0)	True 반환
inday ('01/12/2006 12:23:00 PM', '01/13/2006 12:00:00 AM', 0)	False 반환
inday ('01/12/2006 12:23:00 PM', '01/12/2006 12:00:00 AM', -1)	False 반환
inday ('01/11/2006 12:23:00 PM', '01/12/2006 12:00:00 AM', -1)	True 반환
inday ('01/12/2006 12:23:00 PM', '01/12/2006 12:00:00 AM', 0, 0.5)	False 반환
inday ('01/12/2006 11:23:00 AM', '01/12/2006 12:00:00 AM', 0, 0.5)	True 반환

예 1 – LOAD 문(스크립트)

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Transactions라는 테이블에 로드되는 타임스탬프별 트랜잭션을 포함하는 데이터 집합.
- TimeStamp 시스템 변수 (M/D/YYYY h:mm:ss[.fff] TT) 형식으로 제공되는 날짜 필드.
- inday() 필드로 설정된 in_day 함수를 포함하는 선행 LOAD입니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- in_day

결과 테이블

날짜	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

in_day 필드는 inday() 함수를 사용하고 날짜 필드, 1월 5일에 대한 하드 코딩된 타임스탬프 및 period_no에 0을 함수의 인수로 전달하여 이전 LOAD 문에서 만들어집니다.

예 2 – period_no

로드 스크립트 및 결과

개요

로드 스크립트는 첫 번째 예에서 사용한 것과 동일한 데이터 집합 및 시나리오를 사용합니다.

그러나 이 예에서 작업은 트랜잭션 날짜가 1월 5일 2일 이전에 발생했는지 여부를 계산하는 것입니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- in_day

결과 테이블

날짜	in_day
01/01/2022 7:34:46 PM	0
01/01/2022 10:10:22 PM	0
01/02/2022 8:35:54 AM	0
01/03/2022 2:21:53 PM	-1
01/04/2022 6:49:38 PM	0
01/04/2022 10:58:34 PM	0
01/05/2022 5:40:49 AM	0
01/05/2022 11:29:38 AM	0
01/05/2022 7:04:57 PM	0
01/06/2022 8:49:09 AM	0

이 경우 inday() 함수에서 오프셋 인수로 period_no에 -2를 사용했으므로 함수는 각 트랜잭션 날짜가 1월 3일에 발생했는지 여부를 확인합니다. 이는 한 트랜잭션이 TRUE의 부울 결과를 반환하는 출력 테이블에서 확인할 수 있습니다.

예 3 – day_start

로드 스크립트 및 결과

개요

로드 스크립트는 이전 예에서 사용한 것과 동일한 데이터 집합 및 시나리오를 사용합니다.

그러나 이 예에서 회사 정책은 근무일이 오전 7시에 시작되고 끝나는 것입니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- in_day

결과 테이블

날짜	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

날짜	in_day
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

start_day 인수가 7/24(오전 7시)로 inday() 함수에 사용되므로 이 함수는 각 트랜잭션 날짜가 1월 4일 오전 7시와 1월 5일 오전 7시 이전에 발생했는지 여부를 확인합니다.

이는 1월 4일 오전 7시 이후에 발생한 트랜잭션이 TRUE의 부울 결과를 반환하고 1월 5일 오전 7시 이후에 발생한 트랜잭션이 FALSE의 부울 결과를 반환하는 출력 테이블에서 확인할 수 있습니다.

예 4 – 차트 개체

로드 스크립트 및 차트 표현식

개요

로드 스크립트는 이전 예에서 사용한 것과 동일한 데이터 집합 및 시나리오를 사용합니다.

그러나 이 예에서 데이터 집합은 변경되지 않고 응용 프로그램에 로드됩니다. 차트 개체에서 측정값을 만들어 트랜잭션이 1월 5일에 발생하는지 확인하기 위해 계산합니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다.

- date

트랜잭션이 1월 5일에 발생했는지 여부를 계산하려면 다음 측정값을 만듭니다.

```
=inday(date,'01/05/2022 12:00:00 AM',0)
```

결과 테이블

날짜	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

예 5 – 시나리오

로드 스크립트 및 결과

개요

이 예에서는 1월 5일에 제조된 제품이 장비 오류로 인해 결함이 있는 것으로 확인되었습니다. 최종 사용자는 제조된 제품이 '결함' 또는 '무결함'인 상태와 1월 5일에 제조된 제품의 비용을 날짜별로 표시하는 차트 개체를 원합니다.

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 'Products'라는 테이블에 로드되는 데이터 집합입니다.
- 테이블에는 다음 필드가 포함됩니다.
    - 제품 ID
    - 제조 시간
    - 가격

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다.

```
=dayname(manufacture_date)
```

다음 측정값을 만듭니다.

- `=if(only(InDay(manufacture_date,makedate(2022,01,05),0)),'Defective','Faultless')`
- `=sum(cost_price)`

측정값의 **숫자 형식**을 **화폐**로 설정합니다.

모양에서 **합계**를 끕니다.

결과 테이블

dayname (manufacture_date)	**=if(only(InDay(manufacture_date,makedate (2022,01,05),0)),'결함','무결함')**	**=sum(cost_ price)**
01/01/2022	무결함	44.67
01/02/2022	무결함	36.34
01/03/2022	무결함	51.75
01/04/2022	무결함	89.69
01/05/2022	결함	170.78
01/06/2022	무결함	74.23

`inday()` 함수는 각 제품의 제조 날짜를 평가할 때 부울 값을 반환합니다. 1월 5일에 제조된 모든 제품의 경우 `inday()` 함수는 부울 값 TRUE를 반환하고 제품을 '결함'으로 표시합니다. FALSE 값을 반환하여 당일 제조되지 않은 제품의 경우 제품을 '무결함'으로 표시합니다.

indaytotime

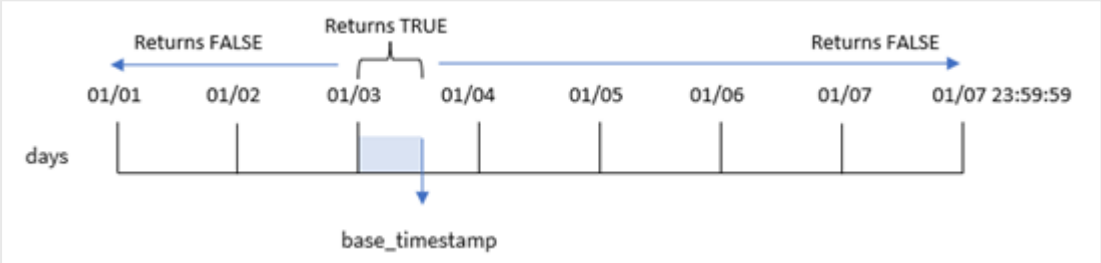
이 함수는 **timestamp**가 **base_timestamp**의 정확한 밀리초까지 포함하여 **base_timestamp**를 포함한 날의 일부에 속할 경우 True를 반환합니다.

구문:

```
InDayToTime (timestamp, base_timestamp, period_no[, day_start])
```

indaytotime() 함수는 하루의 세그먼트 동안 타임스탬프 값이 발생하는 시기에 따라 부울 결과를 반환합니다. 이 세그먼트의 시작 경계는 기본적으로 자정으로 설정되는 날짜의 시작입니다. 날짜의 시작은 indaytotime() 함수의 day_start 인수로 수정할 수 있습니다. 하루 세그먼트의 끝 경계는 함수의 base_timestamp 인수에 의해 결정됩니다.

indaytotime 함수 다이어그램.



사용 시기

indaytotime() 함수는 부울 결과를 반환합니다. 일반적으로 이 유형의 함수는 if expression의 조건으로 사용됩니다. indaytotime() 함수는 기준 타임스탬프의 시간을 포함하여 해당 날짜의 세그먼트에서 타임스탬프가 발생했는지 여부에 따라 집계 또는 계산을 반환합니다.

예를 들어, indaytotime() 함수를 사용하여 오늘까지 발생한 상영의 티켓 판매 합계를 표시할 수 있습니다.

반환 데이터 유형: 부울

Qlik Sense에서 부울 true 값은 -1로 표시되고 false 값은 0으로 표시됩니다.

인수

인수	설명
timestamp	base_timestamp와 비교할 날짜 및 시간입니다.
base_timestamp	타임스탬프를 평가하는 데 사용되는 날짜와 시간입니다.
period_no	일은 period_no로 오프셋을 지정할 수 있습니다. period_no는 정수이며, 값 0은 base_timestamp를 포함하는 날을 나타냅니다. period_no가 음수 값일 경우 이전 날, 양수 값일 경우 이후 날을 나타냅니다.
day_start	(선택 사항) 작업일이 자정부터 시작되지 않도록 하려면 day_start에 일의 분위수 형식으로 오프셋을 지정하십시오. 예를 들어 0.125를 사용하여 오전 3시를 나타냅니다.

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

함수 예

예	결과
indaytotime ('01/12/2006 12:23:00 PM', '01/12/2006 11:59:00 PM', 0)	True 반환
indaytotime ('01/12/2006 12:23:00 PM', '01/12/2006 12:00:00 AM', 0)	False 반환
indaytotime '01/11/2006 12:23:00 PM', '01/12/2006 11:59:00 PM', -1)	True 반환

예 1 – 추가 인수 없음

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 1월 4일에서 5일 사이의 기간에 대한 트랜잭션 집합을 포함하는 데이터 집합이 'Transactions'라는 테이블에 로드됩니다.
- TimeStamp 시스템 변수 (M/D/YYYY h:mm:ss[.fff] TT) 형식으로 제공되는 날짜 필드.
- 각 트랜잭션이 오전 9시 이전에 발생하는지 여부를 결정하는 'in_day_to_time' 필드로 설정된 indaytotime() 함수를 포함하는 선행 LOAD입니다.

로드 스크립트

```
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
];
```

결과

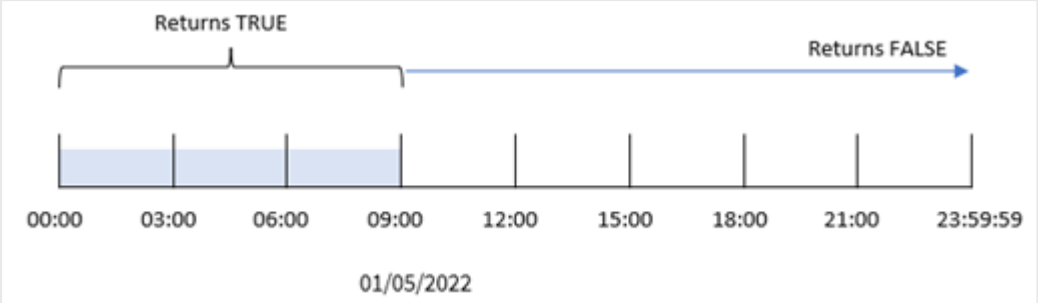
데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- in_day_to_time

결과 테이블

날짜	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

예 1 오전 9시 제한이 있는 indaytotime 함수의 다이어그램..



in_day_to_time field는 indaytotime() 함수를 사용하고 날짜 필드, 1월 5일 오전 9시에 대한 하드 코딩된 타임스탬프 및 함수의 인수로 오프셋 0을 전달하여 이전 LOAD 문에서 만들어졌습니다. 1월 5일 자정과 오전 9시 사이에 발생한 모든 트랜잭션은 TRUE를 반환합니다.

예 2 – period_no

로드 스크립트 및 결과

개요

로드 스크립트는 첫 번째 예에서 사용한 것과 동일한 데이터 집합 및 시나리오를 사용합니다.

그러나 이 예에서는 트랜잭션 날짜가 1월 5일 오전 9시 이전에 발생했는지 여부를 계산합니다.

로드 스크립트

```
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
];
```

결과

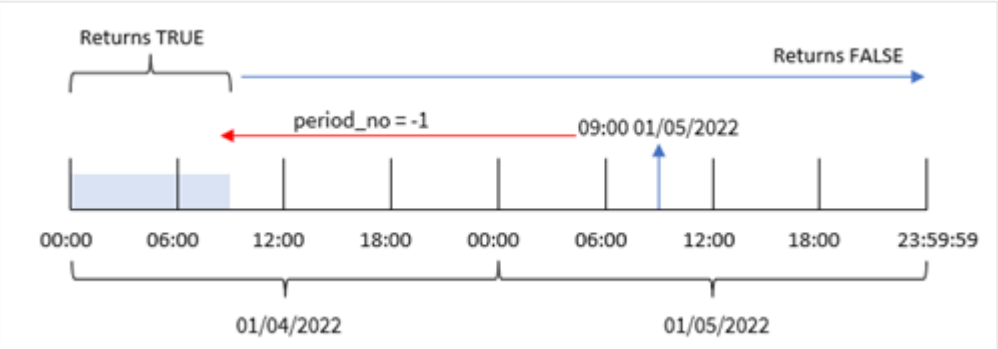
데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- in_day_to_time

결과 테이블

날짜	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

예 2 1월 4 일의 트랜잭션이 있는 indaytotime 함수 다이어그램.



이 예에서는 -1의 오프셋이 indaytotime() 함수의 오프셋 인수로 사용되었으므로 함수는 각 트랜잭션 날짜가 1월 4일 오전 9시 이전에 발생했는지 확인합니다. 이는 트랜잭션이 TRUE의 부울 결과를 반환하는 출력 테이블에서 확인할 수 있습니다.

예 3 – day_start

로드 스크립트 및 결과

개요

첫 번째 예와 동일한 데이터 집합 및 시나리오가 사용됩니다.

그러나 이 예에서 회사 정책은 근무일이 오전 8시에 시작되고 끝나는 것입니다.

로드 스크립트

```
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
];
```

결과

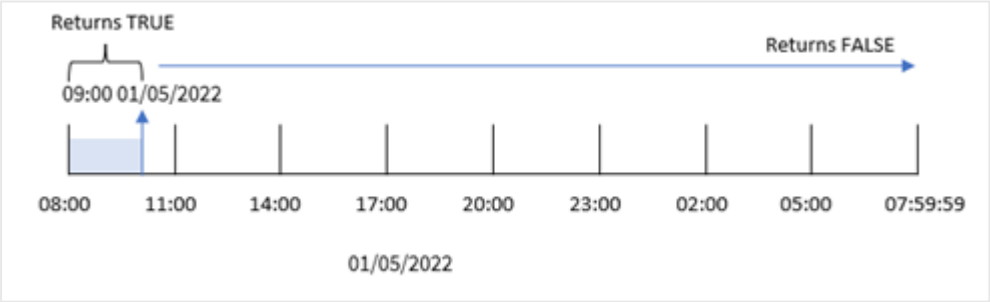
데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- in_day_to_time

결과 테이블

날짜	in_day_to_time
01/04/2022 3:41:54 AM	0
01/04/2022 4:19:43 AM	0
01/04/2022 04:53:47 AM	0
01/04/2022 8:38:53 AM	0
01/04/2022 10:37:52 AM	0
01/04/2022 1:54:10 PM	0
01/04/2022 5:53:23 PM	0
01/04/2022 8:13:26 PM	0
01/04/2022 10:00:49 PM	0
01/05/2022 7:45:37 AM	0
01/05/2022 8:44:36 AM	-1
01/05/2022 11:26:08 AM	0
01/05/2022 6:43:08 PM	0
01/05/2022 10:54:10 PM	0
01/05/2022 11:09:09 PM	0

예 3 오전 8시부터 오전 9시까지 트랜잭션이 있는 indaytotime 함수 다이어그램,



indaytotime() 함수에서 start_day 인수에 8/24(오전 8시에 해당)을 사용하기 때문에 매일 오전 8시에 시작하고 끝납니다. 따라서 indaytotime() 함수는 1월 5일 오전 8시에서 오전 9시 사이에 발생한 모든 트랜잭션에 대해 부울 결과 TRUE를 반환합니다.

예 4 – 차트 개체

로드 스크립트 및 차트 표현식

개요

첫 번째 예와 동일한 데이터 집합 및 시나리오가 사용됩니다.

그러나 이 예에서 데이터 집합은 변경되지 않고 응용 프로그램에 로드됩니다. 차트 개체에서 측정값을 만들어 트랜잭션이 1월 5일 오전 9시 이전에 발생하는지 확인하기 위해 계산합니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다.

date에서 관리합니다.

트랜잭션이 1월 5일 오전 9시 이전에 발생하는지 확인하려면 다음 측정값을 만듭니다.

```
=indaytotime(date,'01/05/2022 9:00:00 AM',0)
```

결과 테이블

날짜	=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

날짜	=indaytotime(date,'01/05/2022 9:00:00 AM',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

in_day_to_time 측정값은 indaytotime() 함수를 사용하고 날짜 필드, 1월 5일 오전 9시에 대한 하드 코딩된 타임스탬프 및 오프셋 0을 함수의 인수로 전달하여 차트 개체에서 만들어집니다. 1월 5일 자정과 오전 9시 사이에 발생한 모든 트랜잭션은 TRUE를 반환합니다. 이는 결과 테이블에서 검증됩니다.

예 5 – 시나리오

로드 스크립트 및 결과

개요

이 예에서 지역 영화관에 대한 티켓 판매가 포함된 데이터 집합은 Ticket_Sales라는 테이블에 로드됩니다. 오늘은 2022년 5월 3일 오전 11시입니다.

사용자는 KPI 차트 개체에 오늘까지 발생한 모든 상영에서 얻은 매출을 표시하기를 원합니다.

로드 스크립트

```
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
];
```

결과

다음과 같이 하십시오.

1. KPI 개체를 만듭니다.
2. indaytotime() 함수를 사용하여 오늘까지 발생한 상영의 모든 티켓 판매 합계를 표시하는 측정값을 만듭니다.

```
=sum(if(indaytotime([show time],'05/03/2022 11:00:00 AM',0),[ticket price],0))
```

3. KPI 개체 '현재 매출'에 대한 레이블을 만듭니다.
4. 측정값의 **숫자 형식**을 **화폐**로 설정합니다.

2022년 5월 3일 오전 11시까지의 티켓 판매 총액은 $52.50입니다.

indaytotime () 함수는 각 티켓 판매의 상영 시간을 현재 시간('2022/05/03 오전 11:00:00')과 비교할 때 부울 값을 반환합니다. 5월 3일 오전 11시 이전의 모든 상영의 경우 indaytotime () 함수는 TRUE의 부울 값을 반환하고 해당 티켓 가격은 합계에 포함됩니다.

inlunarweek

이 함수는 **timestamp**가 **base_date**를 포함하는 음력 주 안에 있는지 확인합니다. Qlik Sense에서 음력 주는 1월 1일을 주의 첫 번째 날로 계산하여 정의되며, 연도의 마지막 주를 제외하고 각 주는 정확히 7일을 포함합니다.

구문:

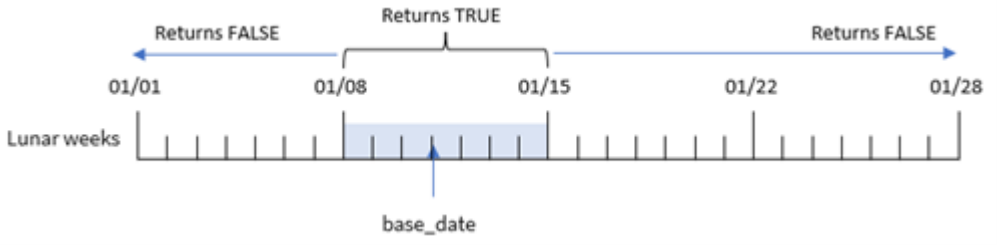
```
InLunarWeek (timestamp, base_date, period_no[, first_week_day])
```

반환 데이터 유형: 부울

Qlik Sense에서 부울 true 값은 -1로 표시되고 false 값은 0으로 표시됩니다.

inlunarweek() 함수는 base_date가 속하는 음력 주를 확인합니다. 그런 다음 각 타임스탬프 값이 base_date와 같은 음력 주에 발생하는지 여부를 확인하면 부울 결과를 반환합니다.

inlunarweek() 함수 다이어그램



사용 시기

inlunarweek() 함수는 부울 결과를 반환합니다. 일반적으로 이 유형의 함수는 IF 표현식의 조건으로 사용됩니다. 이는 평가된 날짜가 해당 음력 주에 발생했는지 여부에 따라 집계 또는 계산을 반환합니다.

예를 들어, inlunarweek() 함수를 사용하여 특정 음력 주에 제조된 모든 장비를 식별할 수 있습니다.

인수

인수	설명
timestamp	**base_date**와 비교할 날짜입니다.
base_date	음력 주를 평가하는 데 사용되는 날짜입니다.
period_no	음력 주는 **period_no**로 오프셋을 지정할 수 있습니다. period_no는 정수이며, 값 0은 **base_date**를 포함하는 음력 주를 나타냅니다. **period_no**가 음수 값일 경우 이전 음력 주, 양수 값일 경우 다음 음력 주를 나타냅니다.
first_week_day	0보다 크거나 작을 수 있는 오프셋입니다. 이 함수는 지정된 일수 및/또는 일의 분위수에 따른 연도 시작 날짜를 변경합니다.

함수 예

예	결과
inlunarweek ('01/12/2013', '01/14/2013', 0)	timestamp의 값인 01/12/2013이 01/08/2013 ~ 01/14/2013 주에 속하므로 TRUE를 반환합니다.
inlunarweek ('01/12/2013', '01/07/2013', 0)	base_date 01/07/2013은 01/01/2013 ~ 01/07/2013으로 정의된 음력 주이므로 FALSE를 반환합니다.
inlunarweek ('01/12/2013', '01/14/2013', -1)	FALSE을 반환합니다. period_no 값을 -1로 지정하면 주가 이전 주, 즉 01/01/2013 ~ 01/07/2013으로 시프트됩니다.
inlunarweek ('01/07/2013', 01/14/2013', -1)	TRUE을 반환합니다. 이전 예와 비교하여 timestamp는 역방향 시프트를 고려한 후의 다음 주입니다.
inlunarweek ('01/11/2006', '01/08/2006', 0, 3)	FALSE을 반환합니다. first_week_day에 3의 값을 지정하면 01/04/2013에서 연도의 시작이 계산됨을 의미합니다. 따라서 base_date의 값은 첫 주에 포함되며, timestamp의 값은 01/11/2013 ~ 01/17/2013 주에 포함됩니다.

inlunarweek() 함수는 다음 함수와 함께 사용되는 경우가 많습니다.

관련 함수

함수	상호 작용
lunarweekname (page 826)	이 함수는 입력 날짜가 발생한 연도의 음력 주 수를 확인하는 데 사용됩니다.

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

예 1 – 추가 인수 없음

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Transactions라는 테이블에 로드되는 1월의 트랜잭션 데이터 집합.
- DateFormat 시스템 변수 서식(MM/DD/YYYY)으로 제공된 날짜 필드.

트랜잭션이 1월 10일과 같은 음력 주에 발생했는지 여부를 확인하는 필드 in_lunar_week를 만듭니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

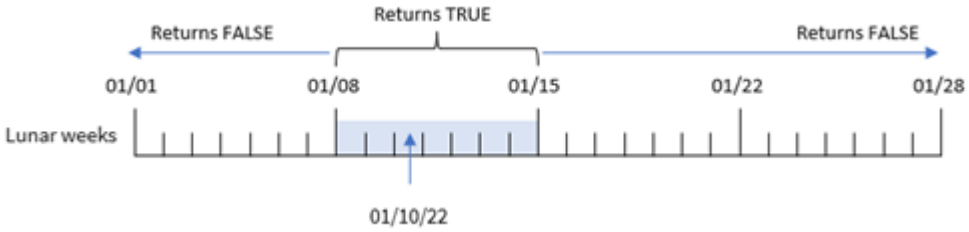
- date
- in_lunar_week

결과 테이블

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

inlunarweek() 함수, 기본 예



in_lunar_week 필드는 inlunarweek() 함수를 사용하여 선행 LOAD 문에서 만들어진 다음 이 함수의 인수로 다음을 전달합니다.

- date 필드
- base_date로 하드 코딩된 1월 10일의 날짜
- 0인 period_no

음력 주는 1월 1일에 시작하므로 1월 10일은 1월 8일에 시작하여 1월 14일에 끝나는 음력 주에 포함됩니다. 따라서 1월의 두 날짜 사이에 발생한 모든 트랜잭션은 TRUE의 부울 값을 반환합니다. 이는 결과 테이블에서 검증됩니다.

예 2 – period_no

예 및 결과:

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 첫 번째 예와 동일한 데이터 집합 및 시나리오.
- DateFormat 시스템 변수 서식(MM/DD/YYYY)으로 제공된 날짜 필드.

그러나 이 예에서 작업은 1월 10일 이후 음력 2주 후에 트랜잭션이 발생했는지 여부를 확인하는 2_lunar_weeks_later 필드를 만드는 것입니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

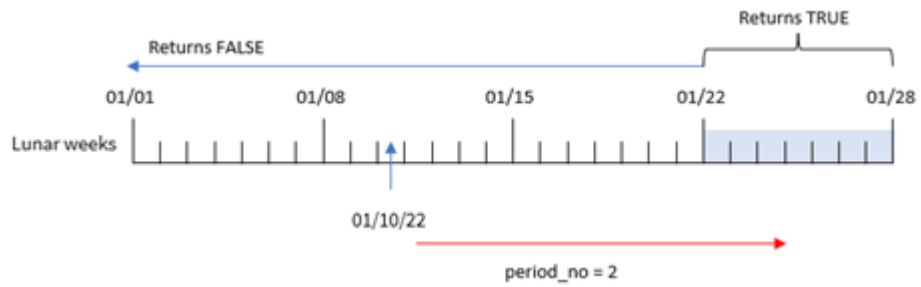
- date
- 2_lunar_weeks_later

결과 테이블

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

inlunarweek() 함수, period_no 예



이 경우 `inlunarweek()` 함수에서 오프셋 인수로 `period_no`가 2로 사용되었으므로 이 함수는 1월 22일부터 시작하는 주를 음력 주로 정의하여 이에 대한 트랜잭션의 유효성을 검사합니다. 따라서 1월 22일과 1월 28일 사이에 발생하는 모든 트랜잭션은 부울 결과 `TRUE`를 반환합니다.

예 3 – first_week_day

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

이 로드 스크립트는 첫 번째 예와 동일한 데이터 집합 및 시나리오를 사용합니다. 그러나 이 예에서는 음력 주를 1월 6일에 시작하도록 설정했습니다.

- 첫 번째 예와 동일한 데이터 집합 및 시나리오.
- 기본 DateFormat 시스템 변수 MM/DD/YYYY가 사용됩니다.
- 5인 first_week_day 인수. 음력 주가 1월 5일에 시작하도록 설정됩니다.

로드 스크립트

```
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
];
```

결과

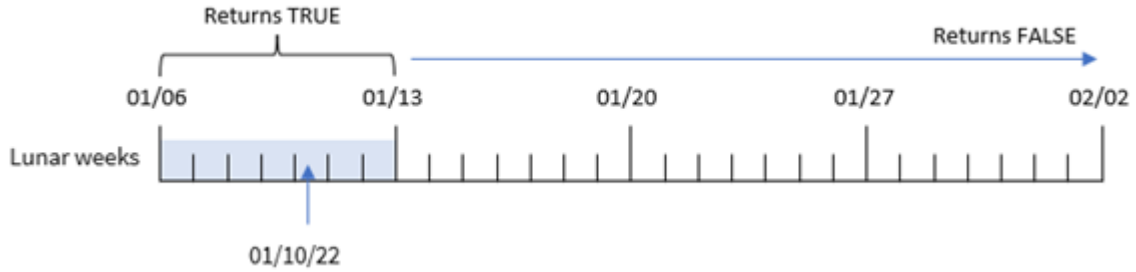
데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- in_lunar_week

결과 테이블

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

inlunarweek() 함수, first_week_day 예



이 경우 inlunarweek() 함수에서 first_week_date 인수가 5로 사용되므로 음력 주 캘린더의 시작을 1월 6일로 오프셋합니다. 따라서 1월 10일은 1월 6일에 시작하여 1월 12일에 끝나는 음력 주에 속합니다. 이 두 날짜 사이에 속한 모든 트랜잭션은 부울 값 TRUE를 반환합니다.

예 4 - 차트 개체

로드 스크립트 및 차트 표현식:

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 첫 번째 예와 동일한 데이터 집합 및 시나리오.
- DateFormat 시스템 변수 서식(MM/DD/YYYY)으로 제공된 날짜 필드.

그러나 이 예에서는 변경되지 않은 데이터 집합이 응용 프로그램에 로드됩니다. 트랜잭션이 1월 10일과 같은 음력 주에 발생했는지 여부를 확인하는 계산은 응용 프로그램의 차트 개체에서 측정값으로 만들어집니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다. date.

트랜잭션이 1월 10일을 포함하는 음력 주에 발생하는지 여부를 계산하려면 다음 측정값을 만듭니다.

```
= inlunarweek(date,'01/10/2022', 0)
```

결과 테이블

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

예 5 - 시나리오

로드 스크립트 및 차트 표현식:

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Products라는 테이블에 로드되는 데이터 집합.
- 제품 ID, 제조 날짜, 원가로 구성된 정보.

장비 오류로 인해 1월 12일을 포함하는 음력 주에 제조된 제품에 결함이 있는 것으로 확인되었습니다. 최종 사용자는 제조된 제품이 '결함' 또는 '무결함'인 상태와 해당 월에 제조된 제품의 비용을 음력 주 이름별로 표시하는 차트 개체를 원합니다.

로드 스크립트

```
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
];
```

결과

다음과 같이 하십시오.

1. 데이터를 로드하고 시트를 엽니다. 새 테이블을 만듭니다.
2. 월 이름을 표시하는 차원을 만듭니다.
   `=lunarweekname(manufacture_date)`
3. `inlunarweek()` 함수를 사용하여 결함이 있는 제품과 결함이 없는 제품을 식별하는 측정값을 만듭니다.
   `=if(only(inlunarweek(manufacture_date,makedate(2022,01,12),0)), 'Defective','Faultless')`
4. 제품의 `cost_price` 합계를 계산하는 측정값을 만듭니다.
   `=sum(cost_price)`
5. 측정값의 **숫자 형식**을 **화폐**로 설정합니다.
6. **모양**에서 **합계**를 끕니다.

결과 테이블

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

inlunarweek() 함수는 각 제품의 제조 날짜를 평가할 때 부울 값을 반환합니다. 1월 10일이 포함된 음력 주에 제조된 모든 제품에 대해 inlunarweek() 함수는 부울 값 TRUE를 반환하고 제품을 '결함'으로 표시합니다. 값 FALSE를 반환하여 해당 주에 제조되지 않은 제품의 경우 제품을 '무결함'으로 표시합니다.

inlunarweektodate

이 함수는 **timestamp**가 **base_date**의 마지막 밀리초까지 포함하여 음력 주의 일부에 속하는지 확인합니다. Qlik Sense에서 음력 주는 1월 1일을 주의 첫 번째 날로 계산하여 정의되며, 연도의 마지막 주를 제외하고 정확히 7일이 포함됩니다.

구문:

```
InLunarWeekToDate (timestamp, base_date, period_no [, first_week_day])
```

반환 데이터 유형: 부울

Qlik Sense에서 부울 true 값은 -1로 표시되고 false 값은 0으로 표시됩니다.

inlunarweektodate() 함수의 다이어그램 예

inlunarweektodate() 함수는 음력 주의 종료 지점 역할을 합니다. 반대로 inlunarweek() 함수는 base_date가 속하는 음력 주를 확인합니다. 예를 들어, base_date가 1월 5일이면 1월 1일과 1월 5일 사이의 모든 타임스탬프는 TRUE 부울 결과를 반환하고 1월 6일과 7일 및 이후 날짜는 FALSE 부울 결과를 반환합니다.

인수

인수	설명
timestamp	**base_date**와 비교할 날짜입니다.
base_date	음력 주를 평가하는 데 사용되는 날짜입니다.

인수	설명
period_no	음력 주는 **period_no**로 오프셋을 지정할 수 있습니다. period_no는 정수이며, 값 0은 **base_date**를 포함하는 음력 주를 나타냅니다. **period_no**가 음수 값일 경우 이전 음력 주, 양수 값일 경우 다음 음력 주를 나타냅니다.
first_week_day	0보다 크거나 작을 수 있는 오프셋입니다. 이 함수는 지정된 일수 및/또는 일의 분위수에 따른 연도 시작 날짜를 변경합니다.

사용 시기

inlunarweektodate() 함수는 부울 결과를 반환합니다. 일반적으로 이 유형의 함수는 IF 표현식의 조건으로 사용됩니다. inlunarweektodate() 함수는 평가 날짜가 해당 주의 특정 세그먼트 동안 발생했는지 여부에 따라 사용자가 계산에서 집계 또는 계산을 반환하기를 원할 때 사용됩니다.

예를 들어, inlunarweektodate() 함수를 사용하여 특정 날짜를 포함하여 특정 주에 제조된 모든 장비를 식별할 수 있습니다.

함수 예

예	결과
inlunarweektodate ('01/12/2013', '01/13/2013', 0)	timestamp의 값인 01/12/2013이 01/08/2013 ~ 01/13/2013 주에 속하므로 TRUE를 반환합니다.
inlunarweektodate ('01/12/2013', '01/11/2013', 0)	두 날짜가 01/12/2012 전의 동일한 음력 주에 속해도 base_date 값이 timestamp 값보다 이후이므로 FALSE를 반환합니다.
inlunarweektodate ('01/12/2006', '01/05/2006', 1)	TRUE을 반환합니다. period_no에 대해 1 값을 지정하면 base_date를 한 주 앞으로 시프트하므로 timestamp 값이 음력 주의 일부에 속하기 때문입니다.

inlunarweektodate() 함수는 다음 함수와 함께 사용되는 경우가 많습니다.

관련 함수

함수	상호 작용
lunarweekname (page 826)	이 함수는 입력 날짜가 발생한 연도의 음력 주 수를 확인하는 데 사용됩니다.

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

예 1 – 추가 인수 없음

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Transactions라는 테이블에 로드되는 1월의 트랜잭션 집합이 포함된 데이터 집합. 기본 DateFormat 시스템 변수 MM/DD/YYYY가 사용됩니다.
- 1월 10일까지 음력 주에 발생한 트랜잭션을 확인하는 필드 in_lunar_week_to_date를 만듭니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- in_lunar_week_to_date

결과 테이블

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

inlunarweektodate() 함수, 추가 인수 없음

in_lunar_week_to_date 필드는 inlunarweektodate() 함수를 사용하고 date 필드, 1월 10일의 하드 코딩된 날짜를 base_date로 전달하고 오프셋 0을 함수의 인수로 전달하여 선행 LOAD 문에서 만들어졌습니다.

음력 주는 1월 1일에 시작하므로 1월 10일은 1월 8일에 시작하는 음력 주에 속합니다. 그리고 inlunarweektodate() 함수를 사용하고 있으므로 해당 음력 주는 10일에 끝납니다. 따라서 1월의 두 날짜 사이에 발생한 모든 트랜잭션은 TRUE의 부울 값을 반환합니다. 이는 결과 테이블에서 검증됩니다.

예 2 – period_no

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

이 로드 스크립트에는 첫 번째 예와 동일한 데이터 집합 및 시나리오가 포함되어 있습니다. 그러나 이 예에서 작업은 음력 주 이후 2주 후부터 1월 1일까지 트랜잭션이 발생했는지 여부를 확인하는 필드 2_lunar_weeks_later를 만드는 것입니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

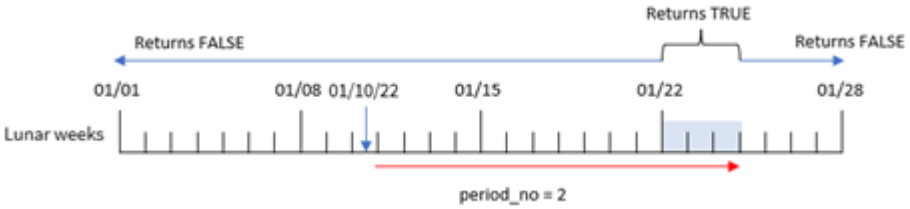
- date
- 2_lunar_weeks_later

결과 테이블

date	2_lunar_weeks_later
1/1/2022	0
1/4/2022	0
1/10/2022	0
1/11/2022	0

date	2_lunar_weeks_later
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

inlunarweektodate() 함수, period_no 예



이 경우 inlunarweektodate() 함수는 1월 10일까지의 음력 주가 3일(1월 8일, 9일, 10일)에 해당한다고 확인합니다. 오프셋 인수로 period_no가 2로 사용되었으므로 이 음력 주는 14일로 이동합니다. 따라서 이는 1월 22일, 23일, 24일이 포함되도록 음력 주 3일을 정의합니다. 1월 22일과 1월 24일 사이에 발생하는 모든 트랜잭션은 부울 결과 TRUE를 반환합니다.

예 3 – first_week_day

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 첫 번째 예와 동일한 데이터 집합 및 시나리오.
- 기본 DateFormat 시스템 변수 MM/DD/YYYY가 사용됩니다.
- 3인 first_week_date 인수. 음력 주가 1월 3일에 시작하도록 설정됩니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

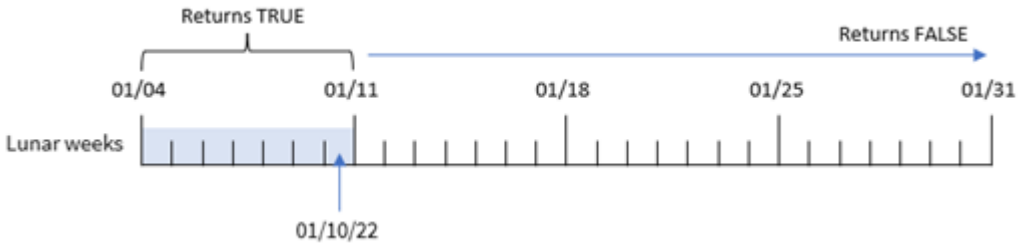
- date
- in_lunar_week_to_date

결과 테이블

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

date	in_lunar_week_to_date
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

inlunarweektodate() 함수, first_week_day 예



이 경우 inlunarweek() 함수에서 3인 the first_week_date 인수를 사용하므로 첫 번째 음력 주는 1월 3일 ~ 1월 10일이 됩니다. 1월 10일도 base_date이므로 이 두 날짜 사이에 있는 모든 트랜잭션은 부울 값 TRUE를 반환합니다.

예 4 - 차트 개체 예

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

이 로드 스크립트에는 첫 번째 예와 동일한 데이터 집합 및 시나리오가 포함되어 있습니다.

그러나 이 예에서는 변경되지 않은 데이터 집합이 응용 프로그램에 로드됩니다. 트랜잭션이 1월 10일까지의 음력 주에 발생했는지 여부를 확인하는 계산은 응용 프로그램의 차트 개체에서 측정값으로 만들어집니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다. date.

다음 측정값을 만듭니다.

```
=inlunarweektodate(date,'01/10/2022', 0)
```

결과 테이블

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
1/15/2022	0
1/17/2022	0
1/18/2022	0
1/19/2022	0
1/21/2022	0
1/23/2022	0

date	=inlunarweektodate(date,'01/10/2022', 0)
1/26/2022	0
1/27/2022	0
1/29/2022	0
1/31/2022	0

inlunarweektodate() 함수, 차트 개체 예

in_lunar_week_to_date 측정값은 inlunarweektodate() 함수를 사용하고 날짜 필드, 1월 10일에 대한 하드 코딩된 날짜를 base_date로 전달하고 오프셋 0을 함수의 인수로 전달하여 차트 개체에서 만들어집니다.

음력 주는 1월 1일에 시작하므로 1월 10일은 1월 8일에 시작하는 음력 주에 속합니다. 또한 inlunarweektodate() 함수를 사용하고 있으므로 해당 음력 주는 10일에 끝납니다. 따라서 1월의 두 날짜 사이에 발생한 모든 트랜잭션은 TRUE의 부울 값을 반환합니다. 이는 결과 테이블에서 검증됩니다.

예 5 - 시나리오

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Products라는 테이블에 로드되는 데이터 집합.
- 제품 ID, 제조 날짜, 원가로 구성된 정보.

장비 오류로 인해 1월 12일 음력 주에 제조된 제품에 결함이 있는 것으로 확인되었습니다. 문제는 1월 13일에 해결되었습니다. 최종 사용자는 제조된 제품이 '결함' 또는 '무결함'인 상태와 해당 주에 제조된 제품의 비용을 주별로 표시하는 차트 개체를 원합니다.

로드 스크립트

```
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
];
```

결과

다음과 같이 하십시오.

1. 데이터를 로드하고 시트를 엽니다. 새 테이블을 만듭니다.
2. 주 이름을 표시할 차원을 만듭니다.
   `=weekname(manufacture_date)`
3. 다음으로, 결함이 있는 제품과 결함이 없는 제품을 식별하는 `inlunarweektodate()` 함수를 사용하는 차원을 만듭니다.
   `=if(inlunarweektodate(manufacture_date,makedate(2022,01,12),0),'Defective','Faultless')`
4. 제품의 `cost_price` 합계를 계산하는 측정값을 만듭니다.
   `=sum(cost_price)`
5. 측정값의 **숫자 형식**을 **화폐**로 설정합니다.

결과 테이블

=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

inlunarweektodate() 함수는 각 제품의 제조 날짜를 평가할 때 부울 값을 반환합니다. 부울 값 TRUE를 반환하는 경우 제품을 'Defective'로 표시합니다. FALSE 값을 반환하는 제품의 경우 1월 12일까지의 음력 주에 제조되지 않은 경우 제품을 'Faultless'로 표시합니다.

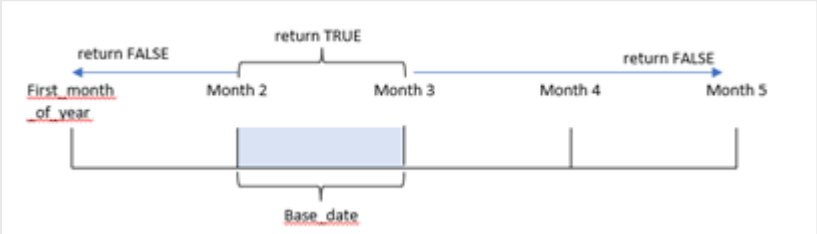
inmonth

timestamp가 **base_date**를 포함하는 월에 속할 경우 이 함수는 True를 반환합니다.

구문:

```
InMonth (timestamp, base_date, period_no)
```

indaytotime 함수 다이어그램.



즉, inmonth() 함수는 날짜 집합이 이 달에 속하는지 여부를 결정하고 해당 월을 식별하는 base_date를 기준으로 부울 값을 반환합니다.

사용 시기

inmonth() 함수는 부울 결과를 반환합니다. 일반적으로 이 유형의 함수는 if expression의 조건으로 사용됩니다. 이는 해당 날짜를 포함하여 날짜가 해당 월에 발생했는지 여부에 따라 집계 또는 계산을 반환합니다.

예를 들어, inmonth() 함수를 사용하여 특정 월에 제조된 모든 장비를 식별할 수 있습니다.

반환 데이터 유형: 부울

Qlik Sense에서 부울 true 값은 -1로 표시되고 false 값은 0으로 표시됩니다.

인수

인수	설명
timestamp	base_date와 비교할 날짜입니다.
base_date	월을 평가하는 데 사용되는 날짜입니다. base_date는 한 달 중 어느 날이든 될 수 있다는 점에 유의해야 합니다.
period_no	월은 period_no로 오프셋을 지정할 수 있습니다. period_no는 정수이며, 값 0은 base_date를 포함하는 월을 나타냅니다. period_no가 음수 값일 경우 이전 달, 양수 값일 경우 다음 달을 나타냅니다.

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

함수 예

예	결과
inmonth ('25/01/2013', '01/01/2013', 0)	True 반환
inmonth('25/01/2013', '23/04/2013', 0)	False 반환
inmonth ('25/01/2013', '01/01/2013', -1)	False 반환
inmonth ('25/12/2012', '17/01/2013', -1)	True 반환

예 1 – 추가 인수 없음

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 2022년 상반기에 대한 트랜잭션 집합이 포함된 데이터 집합.
- 추가 변수 'in_month'를 사용한 선행 LOAD, 트랜잭션이 4월에 발생했는지 여부를 결정합니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- in_month

함수 예

date	in_month
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

'in_month' 필드는 inmonth() 함수를 사용하고, 4월 1일의 하드 코딩된 날짜 필드를 base_date로 전달하고, period_no(0)를 함수의 인수로 전달하여 선행 LOAD 문에서 만들어집니다.

base_date는 TRUE의 부울 결과를 반환할 월을 식별합니다. 따라서 4월에 발생한 모든 트랜잭션은 결과 테이블에서 유효성이 검사된 TRUE를 반환합니다.

예 2 – period_no

로드 스크립트 및 결과

개요

첫 번째 예와 동일한 데이터 집합과 시나리오가 사용됩니다.

그러나 이 예에서는 트랜잭션이 4월의 2개월 전에 발생했는지 여부를 결정하는 '2_months_prior' 필드를 만듭니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- 2_months_prior

함수 예

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

-2를 inmonth() 함수의 period_no 인수로 사용하면 base_date 인수로 정의된 월이 2개월 전으로 이동합니다. 이 예에서는 정의된 월을 4월에서 2월로 변경합니다.

따라서 2월에 발생한 모든 트랜잭션은 TRUE의 부울 결과를 반환합니다.

예 3 – 차트 개체

로드 스크립트 및 차트 표현식

개요

이전 예와 동일한 데이터 집합 및 시나리오가 사용됩니다.

그러나 이 예에서 데이터 집합은 변경되지 않고 응용 프로그램에 로드됩니다. 트랜잭션이 4월에 발생했는지 여부를 결정하는 계산은 응용 프로그램의 차트 개체에서 측정값으로 만들어집니다.

로드 스크립트

```
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
];
```

차트 개체

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다.

```
date
```

트랜잭션이 4월에 발생했는지 여부를 계산하려면 다음 측정값을 만듭니다.

```
=inmonth(date,'04/01/2022', 0)
```

결과

함수 예

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

date	=inmonth(date,'04/01/2022', 0)
6/19/2022	0
6/22/2022	0

예 4 – 시나리오

로드 스크립트 및 결과

개요

이 예에서 데이터 집합은 'Products'라는 테이블에 로드됩니다. 테이블에는 다음 필드가 포함됩니다.

- Product ID
- 제조 날짜
- 가격

2022년 7월에 제조된 제품은 장비 오류로 인해 결함이 있는 것으로 확인되었습니다. 이 문제는 2022년 7월 27일에 해결되었습니다.

최종 사용자는 '결함'(부울 TRUE) 또는 '무결함'(부울 FALSE)으로 제조된 제품의 상태와 해당 월에 제조된 제품의 비용을 월별로 표시하는 차트를 원합니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다.

```
=monthname(manufacture_date)
```

다음 측정값을 만듭니다.

- `=sum(cost_price)`
- `=if(only(inmonth(manufacture_date,makedate(2022,07,01),0)),'Defective','Faultless')`

1. 측정값의 **숫자 형식**을 **화폐**로 설정합니다.
2. **모양**에서 **합계**를 끕니다.

결과 테이블

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
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

`inmonth()` 함수는 각 제품의 제조 날짜를 평가할 때 부울 값을 반환합니다. 2022년 7월에 제조된 모든 제품의 경우 `inmonth()` 함수는 부울 값 True를 반환하고 제품을 'Defective'로 표시합니다. False 값을 반환하여 7월에 제조되지 않은 제품의 경우 제품을 'Faultless'로 표시합니다.

inmonths

이 함수는 타임스탬프가 기준 날짜와 같은 월, 2개월, 분기, 4개월 기간 또는 6개월 내에 속하는지 찾습니다. 또한 타임스탬프가 이전 기간 또는 다음 기간 내에 속하는지도 알아낼 수 있습니다.

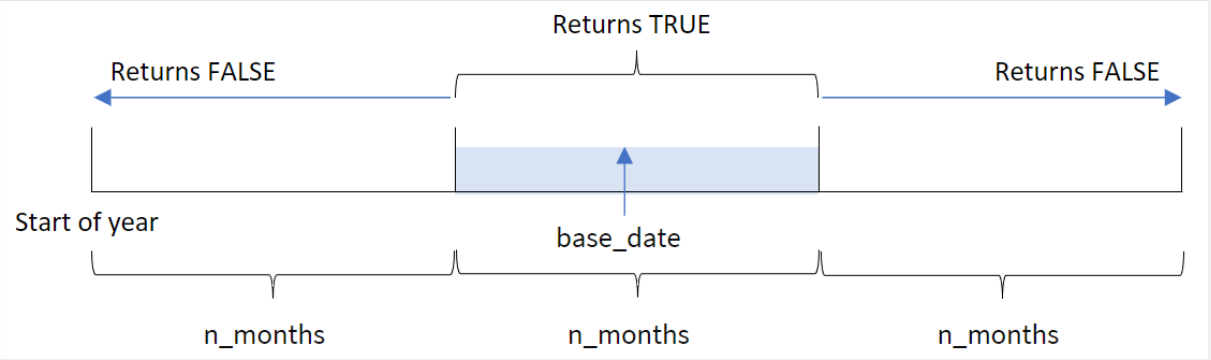
구문:

```
InMonths(n_months, timestamp, base_date, period_no [, first_month_of_year])
```

반환 데이터 유형: 부울

Qlik Sense에서 부울 true 값은 -1로 표시되고 false 값은 0으로 표시됩니다.

inmonths() 함수 다이어그램



inmonths() 함수는 제공된 n_months 인수를 기반으로 연도를 세그먼트로 나눕니다. 그런 다음 평가된 각 타임스탬프가 base_date 인수와 동일한 세그먼트에 속하는지 여부를 확인합니다. 그러나 period_no 인수가 제공되면 함수는 타임스탬프가 base_date에서 이전 기간 또는 다음 기간에 속하는지 여부를 확인합니다.

함수에서 n_month 인수로 사용할 수 있는 연도의 세그먼트는 다음과 같습니다.

n_month 인수

기간	개월 수
1개월	1
2개월	2
분기	3
4개월	4
6개월	6

사용 시기

inmonths() 함수는 부울 결과를 반환합니다. 일반적으로 이 유형의 함수는 if expression의 조건으로 사용됩니다. inmonths() 함수를 사용하여 평가하려는 기간을 선택할 수 있습니다. 예를 들어, 사용자가 특정 기간의 월, 분기 또는 6개월에 제조된 제품을 식별하도록 합니다.

반환 데이터 유형: 부울

Qlik Sense에서 부울 true 값은 -1로 표시되고 false 값은 0으로 표시됩니다.

인수

인수	설명
n_months	기간을 정의하는 개월 수입니다. 정수 또는 정수로 처리되는 표현식으로, 1(inmonth() 함수와 동일), 2(2개월), 3(inquarter() 함수와 동일), 4(4개월 기간) 또는 6(6개월) 중 하나여야 합니다.
timestamp	**base_date**와 비교할 날짜입니다.

인수	설명
base_date	기간을 평가하는 데 사용되는 날짜입니다.
period_no	기간은 **period_no**, 정수 또는 정수로 처리되는 표현식으로 오프셋을 지정할 수 있습니다. 값 0은 **base_date**를 포함하는 기간을 나타냅니다. **period_no**가 음수 값일 경우 이전 기간, 양수 값일 경우 다음 기간을 나타냅니다.
first_month_of_year	1월에 시작되지 않는 (회계)연도를 사용하려는 경우 **first_month_of_year**에 2와 12 사이의 값을 지정하십시오.

다음 값을 사용하여 `first_month_of_year` 인수에서 연도의 첫 번째 달을 설정할 수 있습니다.

first_month_of_year 값

Month	Value
2월	2
3월	3
4월	4
5월	5
6월	6
7월	7
8월	8
9월	9
10월	10
11월	11
12월	12

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 `SET DateFormat` 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

함수 예

예	결과
inmonths(4, '01/25/2013', '04/25/2013', 0)	TRUE를 반환합니다. 타임스탬프 값인 01/25/2013이 base_date 값인 04/25/2013이 속하는 01/01/2013 ~ 04/30/2013의 4개월 기간 내에 포함되기 때문입니다.
inmonths(4, '05/25/2013', '04/25/2013', 0)	False를 반환합니다. 05/25/2013이 이전 예와 동일한 기간에서 벗어나기 때문입니다.
inmonths(4, '11/25/2012', '02/01/2013', -1)	TRUE를 반환합니다. period_no 값인 -1이 검색 기간을 역방향으로 4개월(n-months 값)을 1회 시프트하기 때문으로, 이렇게 하면 검색 기간이 09/01/2012 ~ 12/31/2012로 지정됩니다.
inmonths(4, '05/25/2006', '03/01/2006', 0, 3)	TRUE를 반환합니다. first_month_of_year 값이 3으로 설정되어 있으므로 검색 기간이 01/01/2006 ~ 04/30/2006이 아닌 03/01/2006 ~ 07/30/2006이 됩니다.

예 1 – 추가 인수 없음

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 'Transactions'라는 테이블에 로드되는 2022년 트랜잭션 집합을 포함하는 데이터 집합.
- 추가 변수 'in_months'를 사용한 선행 LOAD, 2022년 5월 15일과 같은 분기에 발생한 트랜잭션을 확인합니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- in_months

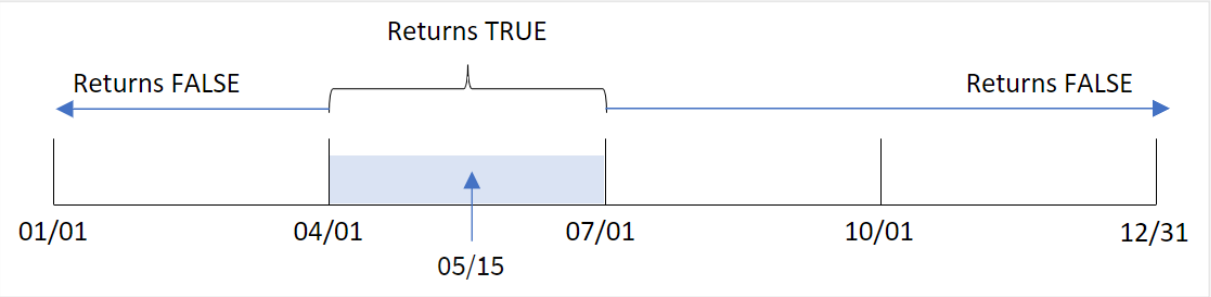
결과 테이블

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

date	in_months
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

'in_months' 필드는 inmonths() 함수를 사용하여 선행 LOAD 문에서 만들어집니다. 제공된 첫 번째 인수는 3이며, 연도를 분기 세그먼트로 나눕니다. 두 번째 인수는 평가 중인 필드(이 예에서는 날짜 필드)를 식별합니다. 세 번째 인수는 5월 15일에 대해 하드 코딩된 날짜인 base_date이고 period_no 0이 마지막 인수입니다.

분기 세그먼트가 있는 inmonths() 함수 다이어그램



5월은 연도의 2분기에 속합니다. 따라서 4월 1일에서 6월 30일 사이에 발생하는 모든 트랜잭션은 부울 결과 TRUE를 반환합니다. 이는 결과 테이블에서 검증됩니다.

예 2 – period_no

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 'Transactions'라는 테이블에 로드되는 2022년 트랜잭션 집합을 포함하는 데이터 집합.
- 추가 변수 'previous_quarter'를 사용한 선행 LOAD, 2022년 5월 15일 이전 분기에 트랜잭션이 발생했는지 여부를 확인합니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- previous_quarter

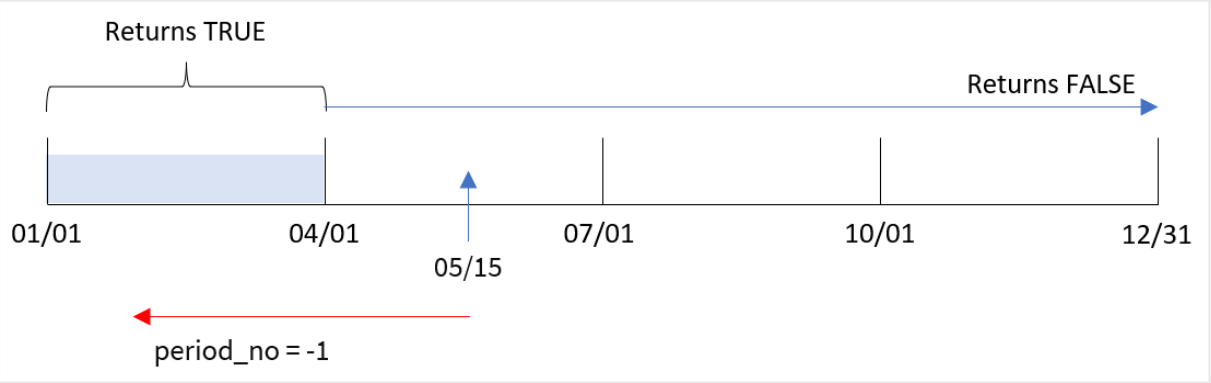
결과 테이블

date	이전 분기
2/19/2022	-1
3/7/2022	-1
3/30/2022	-1
4/5/2022	0
4/16/2022	0
5/1/2022	0
5/7/2022	0
5/22/2022	0

date	이전 분기
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

이 함수는 inmonths() 함수의 period_no 인수로 -1을 사용하여 해당 연도의 1분기에 트랜잭션이 발생했는지 여부를 확인합니다. 5월 15일은 base_date이며 연도의 2분기(4월-6월)에 속합니다.

분기 세그먼트가 있고 period_no가 -1로 설정된 inmonths() 함수 다이어그램



따라서 1월과 3월 사이에 발생하는 모든 트랜잭션은 부울 결과 TRUE를 반환합니다.

예 3 - first_month_of_year

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 'Transactions'라는 테이블에 로드되는 2022년 트랜잭션 집합을 포함하는 데이터 집합.
- 추가 변수 'in_months'를 사용한 선행 LOAD, 2022년 5월 15일과 같은 분기에 발생한 트랜잭션을 확인합니다.

이 예에서 조직 정책은 회계 연도의 첫 번째 달이 되는 3월입니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

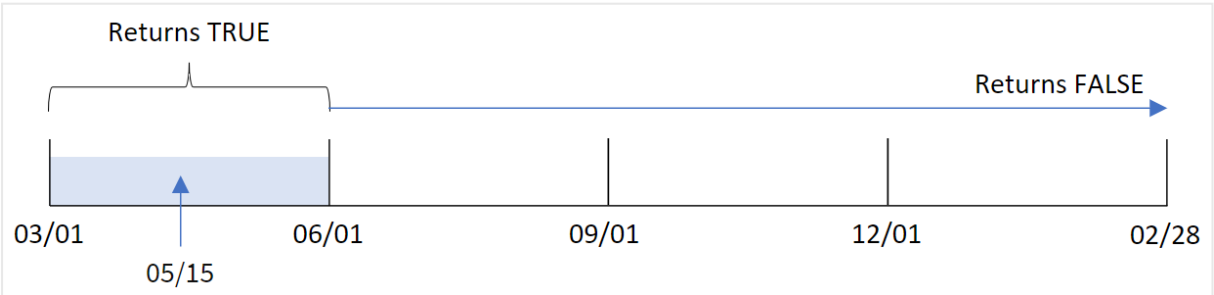
- date
- in_months

결과 테이블

date	in_months
2/19/2022	0
3/7/2022	-1
3/30/2022	-1
4/5/2022	-1
4/16/2022	-1
5/1/2022	-1
5/7/2022	-1
5/22/2022	-1
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

`inmonths()` 함수의 `first_month_of_year` 인수로 3을 사용하면 이 함수는 연도를 3월 1일에 시작합니다. 그런 다음 `inmonths()` 함수는 연도를 다음 분기로 나눕니다. 3월-5월, 6월-8월, 9월-11월, 12월-2월 따라서 5월 15일은 1분기(3월-5월)에 속합니다.

3월을 연도의 첫 번째 달로 설정한 `inmonths()` 함수 다이어그램



해당 월에 발생하는 모든 트랜잭션은 부울 결과 TRUE를 반환합니다.

예 4 - 차트 개체 예

로드 스크립트 및 차트 표현식

개요

첫 번째 예와 동일한 데이터 집합과 시나리오가 사용됩니다.

그러나 이 예에서 데이터 집합은 변경되지 않고 응용 프로그램에 로드됩니다. 2022년 5월 15일과 같은 분기에 트랜잭션이 발생했는지 확인하는 계산이 앱의 차트에 측정값으로 만들어집니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다.

- date

트랜잭션이 5월 15일과 같은 분기에 발생했는지 여부를 계산하려면 다음 측정값을 만듭니다.

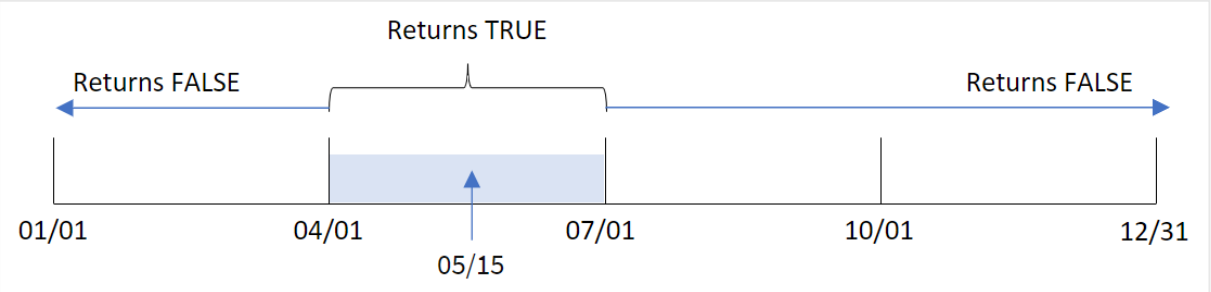
```
=inmonths(3,date,'05/15/2022', 0)
```

결과 테이블

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

'in_months' 필드는 inmonths() 함수를 사용하여 차트에 만들어집니다. 제공된 첫 번째 인수는 3이며, 연도를 분기 세그먼트로 나눕니다. 두 번째 인수는 평가 중인 필드(이 예에서는 날짜 필드)를 식별합니다. 세 번째 인수는 5월 15일에 대해 하드 코딩된 날짜인 base_date이고 period_no 0이 마지막 인수입니다.

분기 세그먼트가 있는 inmonths() 함수 다이어그램



5월은 연도의 2분기에 속합니다. 따라서 4월 1일에서 6월 30일 사이에 발생하는 모든 트랜잭션은 부울 결과 TRUE를 반환합니다. 이는 결과 테이블에서 검증됩니다.

예 5 - 시나리오

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 'Products'라는 테이블에 로드되는 데이터 집합.
- 테이블에는 다음 필드가 포함됩니다.
    - 제품 ID
    - 제품 유형
    - 제조 날짜
    - 가격

최종 사용자는 2021년 첫 번째 세그먼트에서 제조된 제품의 비용을 제품 유형별로 표시하는 차트를 원합니다. 사용자는 이 세그먼트의 길이를 정의할 수 있기를 원합니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다.

로드 스크립트 시작 시 변수 입력 컨트롤에 연결된 변수 vPeriod가 만들어집니다.

다음과 같이 하십시오.

1. 자산 패널에서 **사용자 지정 개체**를 클릭합니다.
2. **Qlik Dashboard 번들**을 선택하고 **변수 입력** 개체를 만듭니다.
3. 차트 개체의 제목을 입력합니다.
4. **변수**에서 이름으로 **vPeriod**를 선택하고 **드롭다운**으로 표시할 개체를 설정합니다.
5. **값**에서 **동적** 값을 클릭합니다. 다음을 입력합니다.
   ='1~month|2~bi-month|3~quarter|4~tertial|6~half-year'.
6. 시트에 새 테이블을 추가합니다.
7. 속성 패널의 **데이터**에서 차원으로 product_type을 추가합니다.
8. 다음 표현식을 측정값으로 추가합니다.
   =sum(if(inmonths($(vPeriod),manufacture_date,makedate(2022,01,01),0),cost_price,0))
9. 측정값의 **숫자 형식**을 **화폐**로 설정합니다.

결과 테이블

product_type	=sum(if(inmonths($(vPeriod),manufacture_date,makedate(2022,01,01),0),cost_price,0))
제품 A	$88.27
제품 B	$37.23
제품 C	$17.17
제품 D	$0.00

inmonths() 함수는 사용자 입력을 해당 인수로 사용하여 연도의 시작 세그먼트 크기를 정의합니다. 이 함수는 각 제품의 제조 날짜를 inmonths() 함수의 두 번째 인수로 전달합니다. 1월 1일을 inmonths() 함수의 세 번째 인수로 사용하면 제조 날짜가 연도의 시작 부분에 해당하는 제품은 부울 값 TRUE를 반환하므로 sum 함수는 해당 제품의 비용을 더합니다.

inmonthstodate

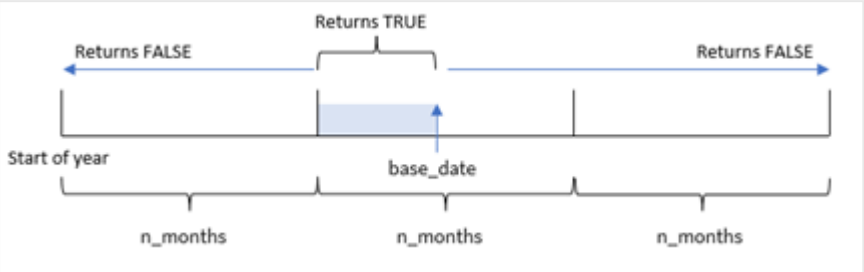
이 함수는 타임스탬프가 base_date의 마지막 밀리초를 포함하여 월, 2개월, 분기, 4개월 기간 또는 6개월 기간에 속하는지 여부를 찾습니다. 또한 타임스탬프가 이전 기간 또는 다음 기간 내에 속하는지도 알아낼 수 있습니다.

구문:

```
InMonths (n_months, timestamp, base_date, period_no[, first_month_of_year ])
```

반환 데이터 유형: 부울

inmonthstodate 함수 다이어그램.



인수

인수	설명
n_months	기간을 정의하는 개월 수입니다. 정수 또는 정수로 처리되는 표현식으로, 1(inmonth() 함수와 동일), 2(2개월), 3(inquarter() 함수와 동일), 4(4개월 기간) 또는 6(6개월) 중 하나여야 합니다.
timestamp	**base_date**와 비교할 날짜입니다.
base_date	기간을 평가하는 데 사용되는 날짜입니다.
period_no	기간은 **period_no**, 정수 또는 정수로 처리되는 표현식으로 오프셋을 지정할 수 있습니다. 값 0은 **base_date**를 포함하는 기간을 나타냅니다. **period_no**가 음수 값일 경우 이전 기간, 양수 값일 경우 다음 기간을 나타냅니다.
first_month_of_year	1월에 시작되지 않는 (회계)연도를 사용하려는 경우 **first_month_of_year**에 2와 12 사이의 값을 지정하십시오.

inmonthstodate() 함수에서 base_date는 특정 연도 세그먼트의 종료 지점 역할을 합니다.

예를 들어, 연도가 3차 세그먼트로 분할되고 base_date가 5월 15일인 경우 1월 초와 4월 말 사이의 타임스탬프는 부울 결과 FALSE를 반환합니다. 5월 1일과 5월 15일 사이의 날짜는 TRUE를 반환합니다. 연도의 나머지 날짜는 FALSE를 반환합니다.

inmonthstodate 함수의 부울 결과 범위 다이어그램.

함수에서 n_month 인수로 사용할 수 있는 연도의 세그먼트는 다음과 같습니다.

n_month 인수

기간	개월 수
1개월	1
2개월	2
분기	3
4개월	4
6개월	6

사용 시기

inmonthstodate() 함수는 부울 결과를 반환합니다. 일반적으로 이 유형의 함수는 if expression의 조건으로 사용됩니다. inmonthstodate() 함수를 사용하여 평가하려는 기간을 선택할 수 있습니다. 예를 들어, 사용자가 특정 날짜까지의 월, 분기 또는 6개월 동안 제조된 제품을 식별할 수 있는 입력 변수를 제공합니다.

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

함수 예

예	결과
inmonthstodate(4, '01/25/2013', '04/25/2013', 0)	timestamp 값인 01/25/2013이 base_date 값인 04/25/2013이 속하는 01/01/2013 ~ 04/25/2013의 4개월 기간 내에 포함되므로 True를 반환합니다.
inmonthstodate(4, '04/26/2013', '04/25/2006', 0)	04/26/2013이 이전 예와 동일한 기간에서 벗어나므로 False를 반환합니다.
inmonthstodate(4, '09/25/2005', '02/01/2006', -1)	True 값인 period_no이 검색 기간을 역방향으로 4개월(-1 값)을 1회 시프트하므로 n-months를 반환합니다. 이렇게 하면 검색 기간이 01/09/2005 ~ 02/01/2006으로 지정됩니다.
inmonthstodate(4, '04/25/2006', '06/01/2006', 0, 3)	first_month_of_year 값을 3으로 설정했으므로 True를 반환합니다. 이렇게 하면 검색 기간이 05/01/2006 ~ 06/01/2006 대신 03/01/2006 ~ 06/01/2006으로 지정됩니다.

예 1 – 추가 인수 없음

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 'Transactions'라는 테이블에 로드되는 2022년 트랜잭션 집합을 포함하는 데이터 집합.
- DateFormat 시스템 변수 (MM/DD/YYYY) 형식의 날짜 필드.
- 다음을 포함하는 선행 LOAD 문:
    - 'in_months_to_date' 필드로 설정된 inmonthstodate() 함수. 이는 2022년 5월 15일까지 분기에 발생한 트랜잭션을 확인합니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- in_months_to_date

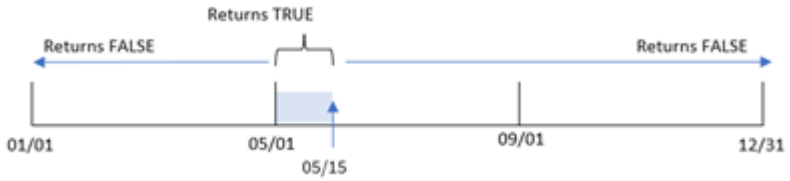
결과 테이블

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
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

'in_months_to_date' 필드는 inmonthstodate() 함수를 사용하여 선행 LOAD 문에서 만들어집니다.

제공된 첫 번째 인수는 3이며, 연도를 분기 세그먼트로 나눕니다. 두 번째 인수는 평가 중인 필드를 식별합니다. 세 번째 인수는 5월 15일의 하드 코딩된 날짜이며 세그먼트의 끝 경계를 정의하는 base_date입니다. period_no 0이 마지막 인수입니다.

추가 인수가 없는 inmonthstodate 함수 다이어그램.



4월 1일과 5월 15일 사이에 발생하는 모든 트랜잭션은 부울 결과 TRUE를 반환합니다. 해당 기간 이외의 트랜잭션 날짜는 FALSE를 반환합니다.

예 2 – period_no

로드 스크립트 및 결과

개요

첫 번째 예와 동일한 데이터 집합 및 시나리오가 사용됩니다.

그러나 이 예에서 작업은 트랜잭션이 5월 15일 이전 분기에 발생했는지 여부를 확인하는 필드 'previous_qtr_to_date'를 만드는 것입니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- previous_qtr_to_date

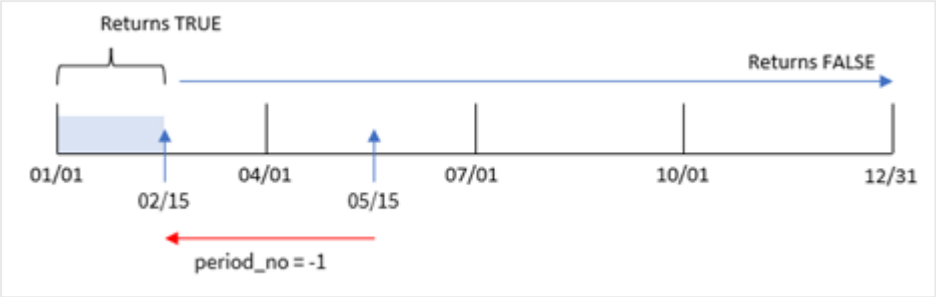
결과 테이블

date	previous_qtr_to_date
1/7/2022	-1
1/19/2022	-1
2/5/2022	-1
2/28/2022	0
3/16/2022	0
4/1/2022	0
5/7/2022	0
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

inmonthstodate() 함수에서 period_no 인수로 -1을 사용하면 이 함수는 비교 연도 세그먼트의 경계를 1/4만큼 시프트합니다.

5월 15일은 연도의 2분기에 속하므로 이 세그먼트는 초기에 4월 1일과 5월 15일 사이에 해당합니다. period_no 인수는 이 세그먼트를 3개월 뒤로 오프셋합니다. 날짜 경계는 1월 1일 ~ 2월 15일입니다.

period_no가 -1로 설정된 inmonthstodate 함수 다이어그램.



따라서 1월 1일과 2월 15일 사이에 발생하는 모든 트랜잭션은 부울 결과 TRUE를 반환합니다.

예 3 – first_month_of_year

로드 스크립트 및 결과

개요

첫 번째 예와 동일한 데이터 집합 및 시나리오가 사용됩니다.

이 예에서 조직 정책은 회계 연도의 첫 번째 달이 되는 3월입니다.

2022년 5월 15일까지 같은 분기에 발생한 트랜잭션을 확인하는 필드 'in_months_to_date'를 만듭니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- in_months_to_date

결과 테이블

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

inmonthstodate() 함수에서 first_month_of_year인수로 3을 사용하면 이 함수는 3월 1일에 연도를 시작한 다음 제공된 첫 번째 인수를 기준으로 연도를 분기로 나눕니다. 따라서 분기 세그먼트는 다음과 같습니다.

- 3월-5월
- 6월-8월
- 9월-11월
- 12월-2월

그런 다음 base_date가 5월 15일이면 끝 경계가 5월 15일로 설정되어 3월-5월 분기가 세그먼트로 분할됩니다.

3월을 연도의 첫 번째 달로 설정한 inmonthstodate 함수 다이어그램.

따라서 3월 1일에서 5월 15일 사이에 발생하는 모든 트랜잭션은 부울 결과 TRUE를 반환하고 날짜가 이 경계를 벗어난 트랜잭션은 FALSE 값을 반환합니다.

예 4 – 차트 예

로드 스크립트 및 차트 표현식

개요

첫 번째 예와 동일한 데이터 집합 및 시나리오가 사용됩니다.

이 예에서 데이터 집합은 변경되지 않고 앱에 로드됩니다. 5월 15일과 같은 분기에 트랜잭션이 발생했는지 여부를 앱 차트의 측정값으로 결정하는 계산을 만드는 작업입니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다.

```
date
```

트랜잭션이 5월 15일과 같은 분기에 발생했는지 여부를 계산하려면 다음 측정값을 만듭니다.

```
=inmonthstodate(3,date,'05/15/2022', 0)
```

결과 테이블

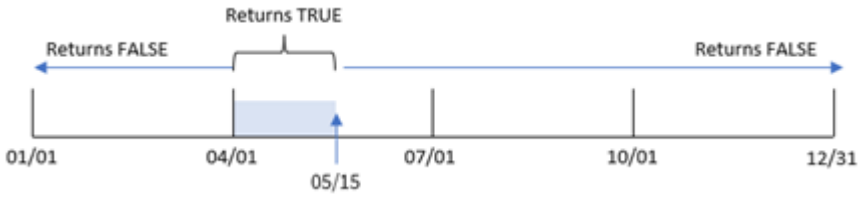
date	=inmonthstodate(3,date,'05/15/2022', 0)
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	0
4/1/2022	-1
5/7/2022	-1
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0

date	=inmonthstodate(3,date,'05/15/2022', 0)
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

'in_months_to_date' 측정값은 inmonthstodate() 함수를 사용하여 차트에 만들어집니다.

제공된 첫 번째 인수는 3이며, 연도를 분기 세그먼트로 나눕니다. 두 번째 인수는 평가 중인 필드를 식별합니다. 세 번째 인수는 하드 코딩된 날짜 5월 15일이며 세그먼트의 끝 경계를 정의하는 base_date입니다. period_no 0이 마지막 인수입니다.

분기 세그먼트가 있는 inmonthstodate 함수 다이어그램.



4월 1일과 5월 15일 사이에 발생하는 모든 트랜잭션은 TRUE의 부울 결과를 반환합니다. 해당 세그먼트 이외의 트랜잭션 날짜는 FALSE를 반환합니다.

예 5 – 시나리오

로드 스크립트 및 결과

개요

이 예에서 데이터 집합은 'Sales'라는 테이블에 로드됩니다. 테이블에는 다음 필드가 포함됩니다.

- Product ID
- Product type
- Sales date
- Sales price

최종 사용자는 2022년 12월 24일까지의 기간에 판매된 제품의 판매량을 제품 유형별로 표시하는 차트를 원합니다. 사용자는 이 기간의 길이를 정의할 수 있기를 원합니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다.

로드 스크립트 시작 시 변수 입력 컨트롤에 연결된 변수 `vPeriod`가 만들어집니다.

다음과 같이 하십시오.

1. 자산 패널에서 **사용자 지정 개체**를 클릭합니다.
2. **Qlik Dashboard 번들**을 선택하고 시트에 **변수 입력**을 추가합니다.
3. 차트의 제목을 입력합니다.
4. **변수**에서 이름으로 **vPeriod**를 선택하고 **드롭다운**으로 표시할 개체를 설정합니다.
5. **값**에서 **동적** 값을 클릭합니다. 다음을 입력합니다.
   `='1~month|2~bi-month|3~quarter|4~tertial|6~half-year'.`
6. 시트에 새 테이블을 추가합니다.
7. 속성 패널의 **데이터**에서 차원으로 `product_type`을 추가합니다.
8. 다음 표현식을 측정값으로 추가합니다.
   `=sum(if(inmonthstodate($(vPeriod),sales_date,makedate(2022,12,24),0),sales_price,0))`
9. 측정값의 **숫자 형식**을 **화폐**로 설정합니다.

결과 테이블

product_type	=sum(if(inmonthstodate($(vPeriod),sales_date,makedate(2022,12,24),0),sales_price,0))
제품 A	$186.48
제품 B	$190.52
제품 C	$220.43
제품 D	$261.46

inmonthstodate() 함수는 사용자 입력을 해당 인수로 사용하여 연도의 시작 세그먼트 크기를 정의합니다.

이 함수는 각 제품의 판매 날짜를 inmonthstodate() 함수의 두 번째 인수로 전달합니다. 12월 24일을 inmonthstodate() 함수의 세 번째 인수로 사용하면 12월 24일까지 정의된 기간에 발생한 판매 날짜가 있는 제품은 부울 값 TRUE를 반환합니다. sum 함수는 이러한 제품의 판매를 더합니다.

inmonthtodate

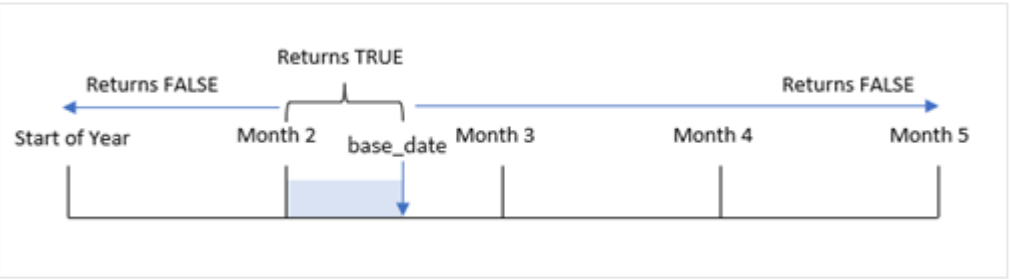
이 함수는 **date**가 **basedate**의 마지막 밀리초까지 포함하여 **basedate**를 포함한 월의 일부에 속할 경우True를 반환합니다.

구문:

```
InMonthToDate (timestamp, base_date, period_no)
```

반환 데이터 유형: 부울

inmonthtodate 함수 다이어그램.



inmonthtodate() 함수는 선택한 월을 세그먼트로 식별합니다. 시작 경계는 월의 시작입니다. 끝 경계는 해당 월의 이후 날짜로 설정할 수 있습니다. 그런 다음 날짜 집합이 이 세그먼트에 속하는지 여부를 확인하여 TRUE 또는 FALSE 부울 값을 반환합니다.

인수

인수	설명
timestamp	**base_date**와 비교할 날짜입니다.
base_date	월을 평가하는 데 사용되는 날짜입니다.
period_no	월은 **period_no**로 오프셋을 지정할 수 있습니다. **period_no**는 정수이며, 값 0은 **base_date**를 포함하는 월을 나타냅니다. **period_no**가 음수 값일 경우 이전 달, 양수 값일 경우 다음 달을 나타냅니다.

사용 시기

inmonthtodate() 함수는 부울 결과를 반환합니다. 일반적으로 이 유형의 함수는 if expression의 조건으로 사용됩니다. inmonthtodate() 함수는 날짜가 해당 날짜를 포함하여 해당 날짜까지 해당 월에 발생했는지 여부에 따라 집계 또는 계산을 반환합니다.

예를 들어, inmonthtodate() 함수를 사용하여 특정 날짜까지 월에 제조된 모든 장비를 식별할 수 있습니다.

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

함수 예

예	결과
inmonthtodate ('01/25/2013', '25/01/2013', 0)	True를 반환합니다.
inmonthtodate ('01/25/2013', '24/01/2013', 0)	False를 반환합니다.
inmonthtodate ('01/25/2013', '28/02/2013', -1)	True를 반환합니다.

예 1 – 추가 인수 없음

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 'Transactions'라는 테이블에 로드되는 2022년 트랜잭션 집합을 포함하는 데이터 집합.
- 날짜 필드가 DateFormat 시스템 변수(MM/DD/YYYY) 형식으로 제공됩니다.
- 다음을 포함하는 선행 LOAD 문:
    - 'in_month_to_date' 필드로 설정되는 inmonthtodate() 함수. 이는 2022년 7월 1일에서 7월 26일 사이에 발생한 트랜잭션을 확인합니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- in_month_to_date

결과 테이블

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

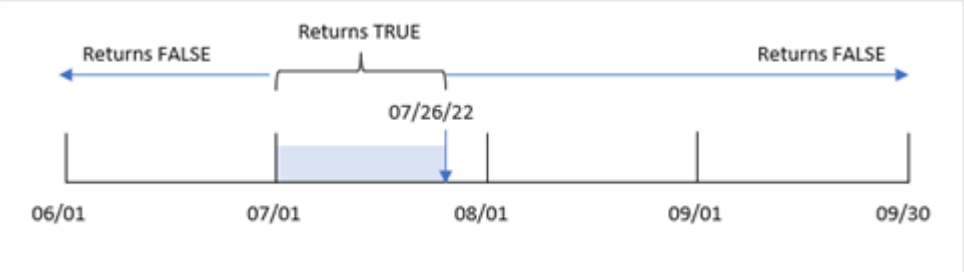
date	in_month_to_date
5/7/2022	0
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	-1
7/22/2022	-1
7/23/2022	-1
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

in_month_to_date 필드는 inmonthtodate() 함수를 사용하여 선행 LOAD 문에서 만들어집니다.

첫 번째 인수는 평가 중인 필드를 식별합니다. 두 번째 인수는 하드 코딩된 날짜인 7월 26일이며 base_date 입니다. 이 base_date 인수는 분할된 월과 해당 세그먼트의 끝 경계를 식별합니다.

period_no 0은 마지막 인수이며, 이는 함수가 세그먼트로 분할된 월 앞이나 뒤에 있는 월을 비교하지 않음을 의미합니다.

추가 인수가 없는 inmonthtodate 함수 다이어그램.



결과적으로 7월 1일과 7월 26일 사이에 발생하는 모든 트랜잭션은 부울 결과 TRUE를 반환합니다. 7월 26일 이후 7월에 발생한 모든 트랜잭션은 해당 연도의 다른 달의 모든 트랜잭션과 마찬가지로 부울 결과 FALSE를 반환합니다.

예 2 – period_no

로드 스크립트 및 결과

개요

첫 번째 예와 동일한 데이터 집합 및 시나리오가 사용됩니다.

이 예에서 작업은 7월 1일과 7월 26일 전에 6개월 동안 발생한 트랜잭션을 확인하는 필드 'six_months_prior'를 만드는 것입니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

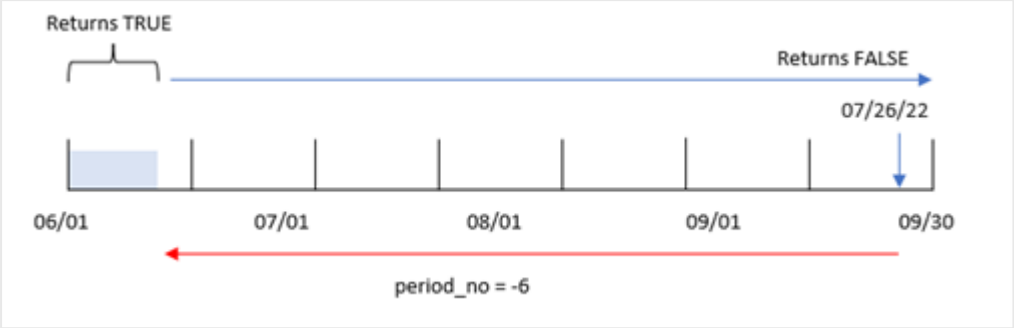
- date
- six_months_prior

결과 테이블

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

`inmonthtodate()` 함수에서 `period_no` 인수로 `-6`을 사용하면 비교 월 세그먼트의 경계가 6개월만큼 시프트합니다. 초기에 월 세그먼트는 7월 1일과 7월 26일 사이에 해당합니다. 그런 다음 `period_no`가 이 세그먼트를 6개월 뒤로 오프셋하면 날짜 경계가 시프트되어 1월 1일과 1월 26일 사이에 포함됩니다.

period_no가 -6으로 설정된 inmonthtodate 함수 다이어그램.



결과적으로 1월 1일과 1월 26일 사이에 발생하는 모든 트랜잭션은 부울 결과 TRUE를 반환합니다.

예 3 – 차트 예

로드 스크립트 및 차트 표현식

개요

첫 번째 예와 동일한 데이터 집합 및 시나리오가 사용됩니다.

이 예에서 데이터 집합은 변경되지 않고 앱에 로드됩니다. 7월 1일부터 7월 26일 사이에 트랜잭션이 발생했는지 여부를 앱 차트의 측정값으로 결정하는 계산을 만드는 작업입니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다.

```
date
```

트랜잭션이 7월 1일과 7월 26일 사이에 발생했는지 여부를 계산하려면 다음 측정값을 만듭니다.

```
=inmonthtodate(date,'07/26/2022', 0)
```

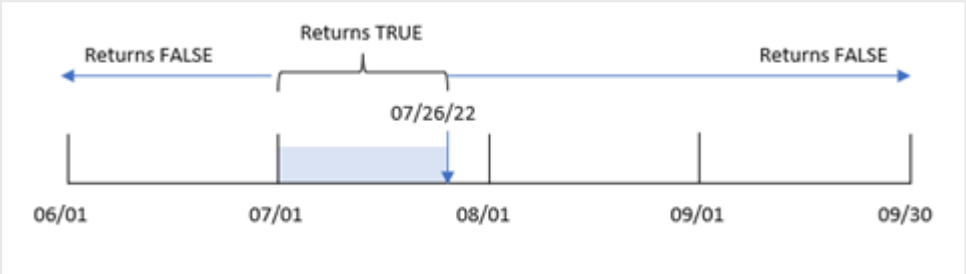
결과 테이블

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

'in_month_to_date' 필드 측정값은 inmonthtodate() 함수를 사용하여 차트에 만들어집니다.

첫 번째 인수는 평가 중인 필드를 식별합니다. 두 번째 인수는 하드 코딩된 날짜인 7월 26일이며 base_date 입니다. 이 base_date 인수는 세그먼트로 분할된 월과 해당 세그먼트의 끝 경계를 식별합니다. period_no 0 은 마지막 인수입니다. 이는 함수가 세그먼트로 분할된 월의 이전 또는 이후 월을 비교하지 않음을 의미합니다.

추가 인수가 없는 inmonthtodate 함수 다이어그램.



결과적으로 7월 1일과 7월 26일 사이에 발생하는 모든 트랜잭션은 부울 결과 TRUE를 반환합니다. 7월 26일 이후 7월에 발생한 모든 트랜잭션은 해당 연도의 다른 달의 모든 트랜잭션과 마찬가지로 부울 결과 FALSE를 반환합니다.

예 4 – 시나리오

로드 스크립트 및 결과

개요

이 예에서 데이터 집합은 'Products'라는 테이블에 로드됩니다. 테이블에는 다음 필드가 포함됩니다.

- Product ID
- 제조 날짜
- 가격

2022년 7월에 제조된 제품은 장비 오류로 인해 결함이 있는 것으로 확인되었습니다. 이 문제는 2022년 7월 27일에 해결되었습니다.

최종 사용자는 '결함'(부울 TRUE) 또는 '무결함'(부울 FALSE)으로 제조된 제품의 상태와 해당 월에 제조된 제품의 비용을 월별로 표시하는 차트를 원합니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- `=monthname(manufacture_date)`
- `=if(Inmonthtodate(manufacture_date,makedate(2022,07,26),0),'Defective','Faultless')`

제품의 총 비용을 계산하려면 다음 측정값을 만듭니다.

```
=sum(cost_price)
```

측정값의 **숫자 형식**을 **화폐**로 설정합니다.

결과 테이블

monthname (manufacture_date)	if(Inmonthtodate(manufacture_date,makedate (2022,07,26),0),'Defective','Faultless')	Sum(cost_ price)
Jan 2022	Faultless	$54.40
Feb 2022	Faultless	$145.69
Mar 2022	Faultless	$53.80
Apr 2022	Faultless	$82.06
May 2022	Faultless	$127.60
Jun 2022	Faultless	$141.82
Jul 2022	Defective	$144.66
Jul 2022	Faultless	$69.98
Aug 2022	Faultless	$147.46
Sep 2022	Faultless	$84.21
Oct 2022	Faultless	$163.91

`inmonthtodate()` 함수는 각 제품의 제조 날짜를 평가할 때 부울 값을 반환합니다.

부울 값 TRUE를 반환하는 날짜의 경우 제품이 'Defective'로 표시됩니다. FALSE 값을 반환하여 7월 26일까지의 월에 제조되지 않은 제품의 경우 제품을 'Faultless'로 표시합니다.

inquarter

timestamp가 **base_date**를 포함하는 분기에 속할 경우 이 함수는 True를 반환합니다.

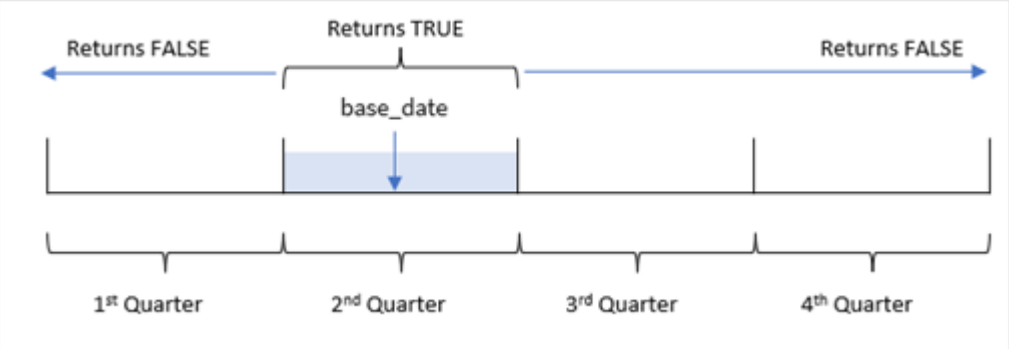
구문:

```
InQuarter (timestamp, base_date, period_no[, first_month_of_year])
```

반환 데이터 유형: 부울

Qlik Sense에서 부울 true 값은 -1로 표시되고 false 값은 0으로 표시됩니다.

inquarter() 함수의 범위 다이어그램



즉, inquarter() 함수는 연도를 1월 1일과 12월 31일 사이의 동일한 4분기로 나눕니다. first_month_of_year 인수를 사용하여 앱에서 첫 번째로 간주되는 월을 변경할 수 있으며 분기는 해당 인수에 따라 변경됩니다. base_date, 이 함수는 함수에 대한 비교 대상으로 사용해야 하는 분기를 식별합니다. 마지막으로 이 함수는 날짜 값을 해당 분기 세그먼트와 비교할 때 부울 결과를 반환합니다.

사용 시기

inquarter() 함수는 부울 결과를 반환합니다. 일반적으로 이 유형의 함수는 if expression의 조건으로 사용됩니다. 이는 선택한 분기에 날짜가 발생했는지 여부에 따라 집계 또는 계산을 반환합니다.

예를 들어, inquarter() 함수를 사용하여 장비가 제조된 날짜를 기준으로 분기 세그먼트로 제조된 모든 장비를 식별할 수 있습니다.

인수

인수	설명
timestamp	**base_date**와 비교할 날짜입니다.
base_date	분기를 평가하는 데 사용되는 날짜입니다.
period_no	분기는 **period_no**로 오프셋을 지정할 수 있습니다. **period_no**는 정수이며, 값 0은 **base_date**를 포함하는 분기를 나타냅니다. **period_no**가 음수 값일 경우 이전 분기, 양수 값일 경우 다음 분기를 나타냅니다.

인수	설명
first_month_of_year	1월에 시작되지 않는 (회계)연도를 사용하려는 경우 **first_month_of_year**에 2와 12 사이의 값을 지정하십시오.

다음 값을 사용하여 `first_month_of_year` 인수에서 연도의 첫 번째 달을 설정할 수 있습니다.

first_month_of_year 값

Month	Value
2월	2
3월	3
4월	4
5월	5
6월	6
7월	7
8월	8
9월	9
10월	10
11월	11
12월	12

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 `SET DateFormat` 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

함수 예

예	결과
inquarter ('01/25/2013', '01/01/2013', 0)	TRUE 반환
inquarter ('01/25/2013', '04/01/2013', 0)	FALSE 반환
inquarter ('01/25/2013', '01/01/2013', -1)	FALSE 반환
inquarter ('12/25/2012', '01/01/2013', -1)	TRUE 반환

예	결과
inquarter ('01/25/2013', '03/01/2013', 0, 3)	FALSE 반환
inquarter ('03/25/2013', '03/01/2013', 0, 3)	TRUE 반환

예 1 – 추가 인수 없음

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 'Transactions'라는 테이블에 로드되는 2022년의 트랜잭션 집합을 포함하는 데이터 집합.
- 'in_quarter' 필드로 설정된 inquarter() 함수를 포함하고 2022년 5월 15일과 같은 분기에 발생한 트랜잭션을 확인하는 선행 LOAD.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

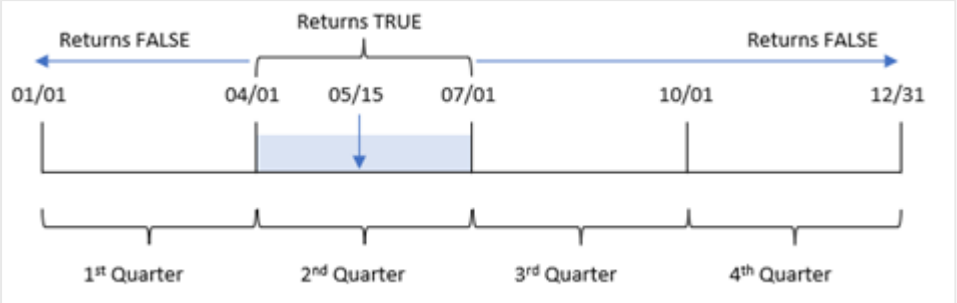
- date
- in_quarter

결과 테이블

date	in_quarter
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	0
4/1/2022	-1
5/7/2022	-1
5/16/2022	-1
6/15/2022	-1
6/26/2022	-1
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

in_quarter 필드는 inquarter() 함수를 사용하여 선행 LOAD 문에서 만들어집니다. 첫 번째 인수는 평가 중인 필드를 식별합니다. 두 번째 인수는 비교 대상으로 정의할 분기를 식별하는 5월 15일의 하드 코딩된 날짜입니다. period_no 0은 마지막 인수이며, 이는 inquarter() 함수가 세그먼트로 분할된 분기 앞이나 뒤에 있는 분기를 비교하지 않습니다.

5월 15일을 기준 날짜로 하는 inquarter() 함수 다이어그램



4월 1일과 6월 30일 사이에 발생하는 모든 트랜잭션은 부울 결과 TRUE를 반환합니다.

예 2 – period_no

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 'Transactions'라는 테이블에 로드되는 2022년의 트랜잭션 집합을 포함하는 데이터 집합.
- 'previous_quarter' 필드로 설정된 inquarter() 함수를 포함하고 2022년 5월 15일의 분기 이전 분기에 발생한 트랜잭션을 확인하는 선행 LOAD.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

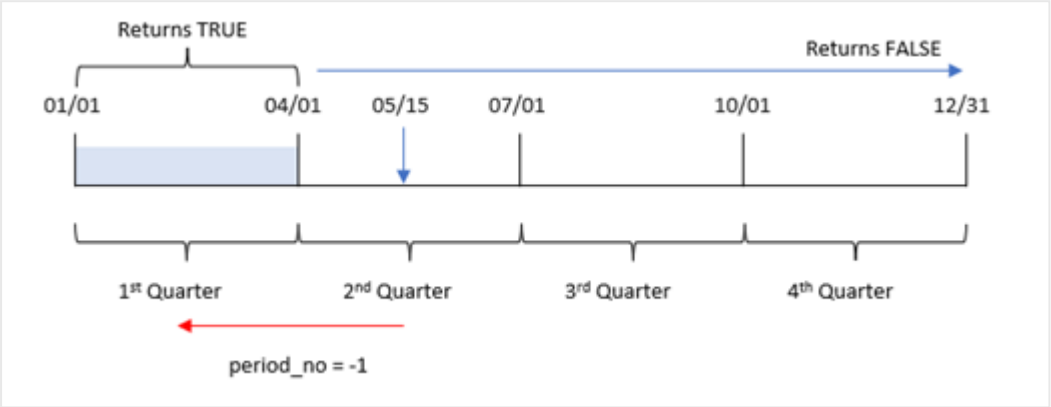
- date
- previous_qtr

결과 테이블

date	previous_qtr
1/7/2022	-1
1/19/2022	-1
2/5/2022	-1
2/28/2022	-1
3/16/2022	-1
4/1/2022	0
5/7/2022	0
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

inquarter() 함수에서 period_no 인수로 -1을 사용하면 비교 대상 분기의 경계가 전체 분기만큼 뒤로 시프트합니다. 5월 15일은 해당 연도의 2분기에 속하므로 이 세그먼트는 초기에 4월 1일부터 6월 30일까지의 분기에 해당합니다. period_no는 이 세그먼트를 3개월 뒤로 오프셋하여 날짜 경계가 1월 1일에서 3월 30일이 되도록 합니다.

5월 15일을 기준 날짜로 하는 inquarter() 함수 다이어그램



따라서 1월 1일에서 3월 30일 사이에 발생하는 모든 트랜잭션은 부울 결과 TRUE를 반환합니다.

예 3 - first_month_of_year

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 'Transactions'라는 테이블에 로드되는 2022년의 트랜잭션 집합을 포함하는 데이터 집합.
- 'in_quarter' 필드로 설정된 inquarter() 함수를 포함하고 2022년 5월 15일과 같은 분기에 발생한 트랜잭션을 확인하는 선행 LOAD.

그러나 이 예에서 조직 정책은 회계 연도의 첫 번째 달이 되는 3월입니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- previous_qtr

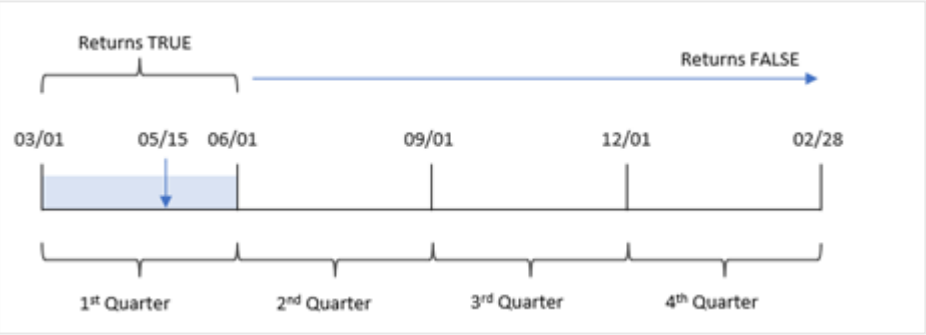
결과 테이블

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

date	previous_qtr
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

inquarter() 함수에서 first_month_of_year 인수로 3을 사용하면 3월 1일이 연도의 시작으로 설정되고 연도를 분기로 나눕니다. 따라서 분기 세그먼트는 3월-5월, 6월-8월, 9월-11월, 12월-2월입니다. 5월 15일의 base_date는 3월~5월 분기를 함수의 비교 대상 분기로 설정합니다.

3월을 연도의 첫 번째 달로 설정한 inquarter() 함수 다이어그램.



따라서 3월 1일과 5월 31일 사이에 발생하는 모든 트랜잭션은 부울 결과 TRUE를 반환합니다.

예 4 - 차트 개체 예

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 'Transactions'라는 테이블에 로드되는 2022년의 트랜잭션 집합을 포함하는 데이터 집합.
- 'in_quarter' 필드로 설정된 inquarter() 함수를 포함하고 2022년 5월 15일과 같은 분기에 발생한 트랜잭션을 확인하는 선행 LOAD.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다.

- date

트랜잭션이 5월 15일과 같은 분기에 발생했는지 여부를 계산하려면 다음 측정값을 만듭니다.

```
=inquarter(date,'05/15/2022', 0)
```

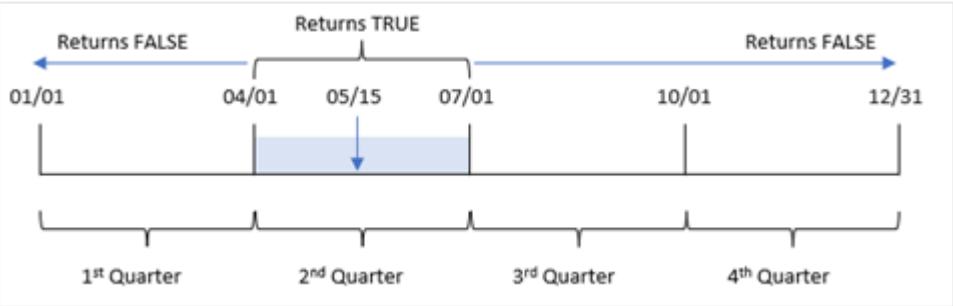
결과 테이블

date	in_quarter
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	0

date	in_quarter
4/1/2022	-1
5/7/2022	-1
5/16/2022	-1
6/15/2022	-1
6/26/2022	-1
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

'in_quarter' 측정값은 inquarter() 함수를 사용하여 차트에 만들어집니다. 첫 번째 인수는 평가 중인 필드를 식별합니다. 두 번째 인수는 비교 대상으로 정의할 분기를 식별하는 5월 15일의 하드 코딩된 날짜입니다. period_no 0은 마지막 인수이며, 이는 inquarter() 함수가 세그먼트로 분할된 분기 앞이나 뒤에 있는 분기를 비교하지 않습니다.

5월 15일을 기준 날짜로 하는 inquarter() 함수 다이어그램



4월 1일과 6월 30일 사이에 발생하는 모든 트랜잭션은 부울 결과 TRUE를 반환합니다.

예 5 - 시나리오

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 'Products'라는 테이블에 로드되는 데이터 집합.
- 테이블에는 다음 필드가 포함됩니다.
    - 제품 ID
    - 제품 유형
    - 제조 날짜
    - 가격

장비 오류로 인해 2022년 5월 15일 분기에 제조된 제품에 결함이 있는 것으로 확인되었습니다. 최종 사용자는 제조된 제품이 '결함' 또는 '무결함'인 상태와 해당 분기에 제조된 제품의 비용을 분기 이름별로 표시하는 차트를 원합니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다.

```
=quartername(manufacture_date)
```

다음 측정값을 만듭니다.

- `=if(only(InQuarter(manufacture_date,makedate(2022,05,15),0)),'Defective','Faultless')`, `inquarter()` 함수를 사용하여 결함이 있는 제품과 결함이 없는 제품을 식별합니다.
- `=sum(cost_price)`, 각 제품의 비용 합계를 표시합니다.

다음과 같이 하십시오.

1. 측정값의 **숫자 형식**을 **화폐**로 설정합니다.
2. **모양**에서 **합계**를 끕니다.

결과 테이블

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

`inquarter()` 함수는 각 제품의 제조 날짜를 평가할 때 부울 값을 반환합니다. 5월 15일이 포함된 분기에 제조된 모든 제품의 경우 `inquarter()` 함수는 부울 값 TRUE를 반환하고 제품을 'Defective'로 표시합니다. FALSE 값을 반환하여 해당 분기에 제조되지 않은 제품의 경우 제품을 'Faultless'로 표시합니다.

inquartertodate

이 함수는 **timestamp**가 **base_date**의 마지막 밀리초까지 포함하여 **base_date**를 포함한 분기의 일부에 속할 경우 True를 반환합니다.

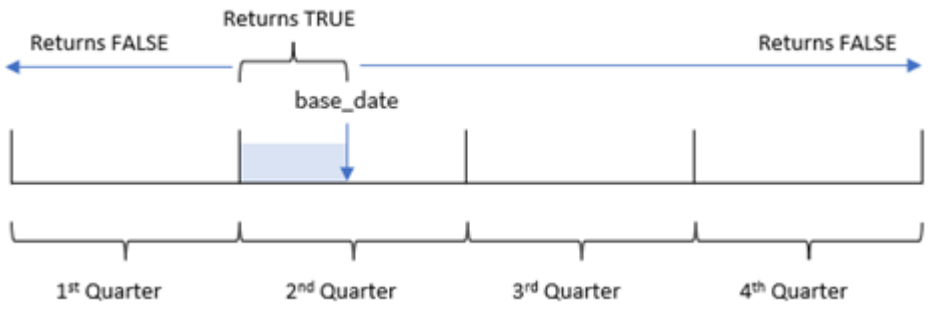
구문:

```
InQuarterToDate (timestamp, base_date, period_no [, first_month_of_year])
```

반환 데이터 유형: 부울

Qlik Sense에서 부울 true 값은 -1로 표시되고 false 값은 0으로 표시됩니다.

inquartertodate 함수의 다이어그램



inquartertodate() 함수는 연도를 1월 1일과 12월 31일(또는 사용자 정의 연도 시작 및 해당 종료 날짜) 사이의 동일한 4분기로 나눕니다. base_date를 사용하면 이 함수가 특정 분기를 분할하고 base_date를 사용하여 해당 분기 세그먼트에 대해 허용되는 최대 날짜와 분기를 식별합니다. 마지막으로 이 함수는 지정된 날짜 값을 해당 세그먼트와 비교할 때 부울 결과를 반환합니다.

인수

인수	설명
timestamp	**base_date**와 비교할 날짜입니다.
base_date	분기를 평가하는 데 사용되는 날짜입니다.
period_no	분기는 **period_no**로 오프셋을 지정할 수 있습니다. **period_no**는 정수이며, 값 0은 **base_date**를 포함하는 분기를 나타냅니다. **period_no**가 음수 값일 경우 이전 분기, 양수 값일 경우 다음 분기를 나타냅니다.
first_month_of_year	1월에 시작되지 않는 (회계)연도를 사용하려는 경우 **first_month_of_year**에 2와 12 사이의 값을 지정하십시오.

사용 시기

inquartertodate() 함수는 부울 결과를 반환합니다. 일반적으로 이 유형의 함수는 if 표현식의 조건으로 사용됩니다. inquartertodate() 함수는 평가된 날짜가 해당 날짜를 포함하여 해당 날짜까지 분기에 발생했는지 여부에 따라 집계 또는 계산을 반환하는 데 사용됩니다.

예를 들어, inquartertodate() 함수를 사용하여 특정 날짜까지 분기에 제조된 모든 장비를 식별할 수 있습니다.

함수 예

예	결과
inquartertodate ('01/25/2013', '03/25/2013', 0)	timestamp 값 01/25/2013이 01/01/2013부터 03/25/2013까지의 3개월 기간 내에 있으므로 TRUE를 반환하고, 이 기간에는 base_date 값 03/25/2013이 있습니다.
inquartertodate ('04/26/2013', '03/25/2013', 0)	04/26/2013이 이전 예와 같은 기간 밖에 있으므로 FALSE를 반환합니다.

예	결과
inquartertodate ('02/25/2013', '06/09/2013', -1)	period_no 값 -1이 3개월(연도의 1/4) 기준의 한 기간 뒤로 검색 기간을 이동하므로 TRUE를 반환합니다. 이렇게 하면 검색 기간이 01/01/2013 ~ 03/09/2013이 됩니다.
inquartertodate ('03/25/2006', '04/15/2006', 0, 2)	first_month_of_year 값이 2로 설정되어 있으므로 TRUE를 반환하며, 검색 기간이 04/01/2006 ~ 04/15/2006가 아닌 02/01/2006 ~ 04/15/2006이 됩니다.

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

예 1 – 추가 인수 없음

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Transactions라는 테이블에 로드되는 2022년 트랜잭션 집합이 포함된 데이터 집합.
- DateFormat 시스템 변수(MM/DD/YYYY) 서식으로 제공된 날짜 필드.
- 2022년 5월 15일까지 분기에 발생한 트랜잭션을 결정하는 필드 in_quarter_to_date 만들기.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- `date`
- `in_quarter_to_date`

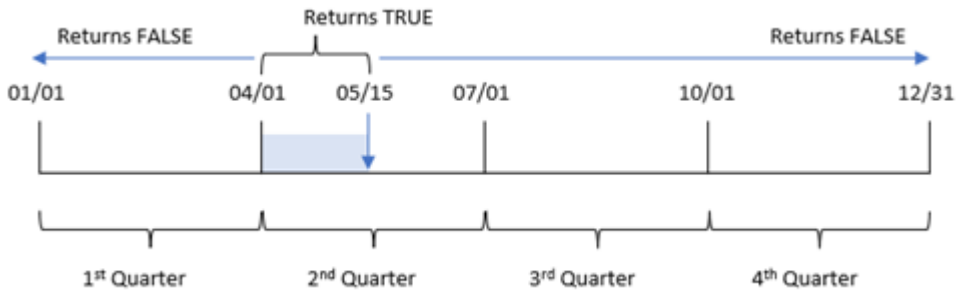
결과 테이블

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

date	in_quarter_to_date
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

'in_quarter_to_date' 필드는 inquartertodate() 함수를 사용하여 선행 LOAD 문에서 만들어집니다. 제공된 첫 번째 인수는 평가 중인 필드를 식별합니다. 두 번째 인수는 5월 15일에 대한 하드 코딩된 날짜이며, 세그먼트로 분할할 분기를 식별하고 해당 세그먼트의 끝 경계를 정의하는 base_date입니다. period_no 0은 마지막 인수이며, 이는 함수가 세그먼트로 분할된 분기 앞이나 뒤에 있는 분기를 비교하지 않음을 의미합니다.

inquartertodate 함수의 다이어그램, 추가 인수 없음



4월 1일과 5월 15일 사이에 발생하는 모든 트랜잭션은 부울 결과 TRUE를 반환합니다. 5월 16일 이후의 트랜잭션 날짜는 4월 1일 이전의 트랜잭션과 마찬가지로 FALSE를 반환됩니다.

예 2 – period_no

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 첫 번째 예와 동일한 데이터 집합 및 시나리오.
- 2022년 5월 15일에 끝나는 분기 세그먼트 이전의 전체 분기에 발생하는 트랜잭션을 결정하는 필드 previous_qtr_to_date 만들기.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- previous_qtr_to_date

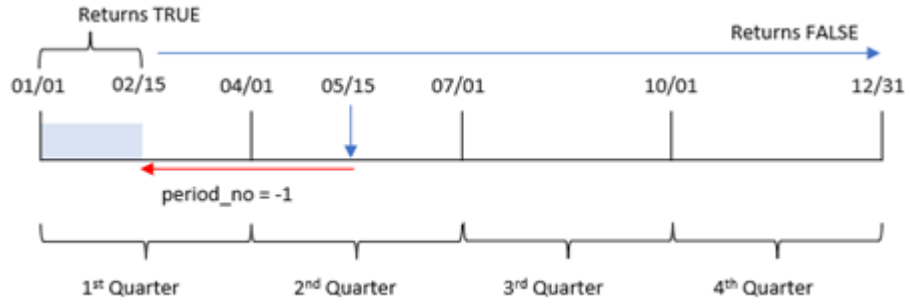
결과 테이블

date	previous_qtr_to_date
1/7/2022	-1
1/19/2022	-1
2/5/2022	-1
2/28/2022	0

date	previous_qtr_to_date
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

period_no 값 -1은 inquartertodate () 함수가 입력 분기 세그먼트를 이전 분기와 비교함을 나타냅니다. 5월 15일은 올해의 2분기에 속하므로 이 세그먼트는 처음에 4월 1일과 5월 15일 사이에 해당합니다. 그런 다음 period_no는 이 세그먼트를 3개월 앞서 오프셋하여 날짜 경계가 1월 1일에서 2월 15일이 되도록 합니다.

inquartertodate 함수의 다이어그램, period_no 예



따라서 1월 1일과 2월 15일 사이에 발생하는 모든 트랜잭션은 부울 결과 TRUE를 반환합니다.

예 3 – first_month_of_year

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 첫 번째 예와 동일한 데이터 집합 및 시나리오.
- 2022년 5월 15일까지 같은 분기에 발생한 트랜잭션을 결정하는 필드 in_quarter_to_date 만들기.

이 예에서는 3월을 회계 연도의 첫 번째 달로 설정합니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- in_quarter_to_date

결과 테이블

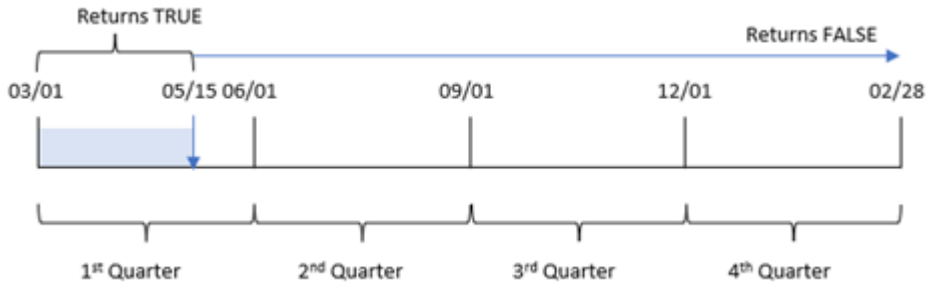
date	in_quarter_to_date
1/7/2022	0
1/19/2022	0
2/5/2022	0
2/28/2022	0
3/16/2022	-1
4/1/2022	-1
5/7/2022	-1
5/16/2022	0
6/15/2022	0
6/26/2022	0
7/9/2022	0
7/22/2022	0
7/23/2022	0
7/27/2022	0
8/2/2022	0
8/8/2022	0
8/19/2022	0
9/26/2022	0
10/14/2022	0
10/29/2022	0

inquartertodate() 함수에서 3을 first_month_of_year 인수로 사용하면 이 함수는 3월 1일에 연도를 시작한 다음 연도를 분기로 나눕니다. 따라서 분기 세그먼트는 다음과 같습니다.

- 3월~5월
- 6월~8월
- 9월~11월
- 12월~2월

그런 다음 base_date가 5월 15일이면 끝 경계가 5월 15일로 설정되어 3월에서 5월까지 분기가 세그먼트로 분할됩니다.

inquartertodate 함수의 다이어그램, first_month_of_year 예



따라서 3월 1일에서 5월 15일 사이에 발생하는 모든 트랜잭션은 부울 결과 TRUE를 반환하는 반면 날짜가 이 경계를 벗어난 트랜잭션은 FALSE 값을 반환합니다.

예 4 – 차트 개체 예

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

이 로드 스크립트에는 첫 번째 예와 동일한 데이터 집합 및 시나리오가 포함되어 있습니다. 그러나 이 예에서는 변경되지 않은 데이터 집합이 응용 프로그램에 로드됩니다. 5월 15일과 같은 분기에 발생한 트랜잭션을 결정하는 계산이 차트 개체의 측정값으로 만들어집니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다.date.

다음 측정값을 만듭니다.

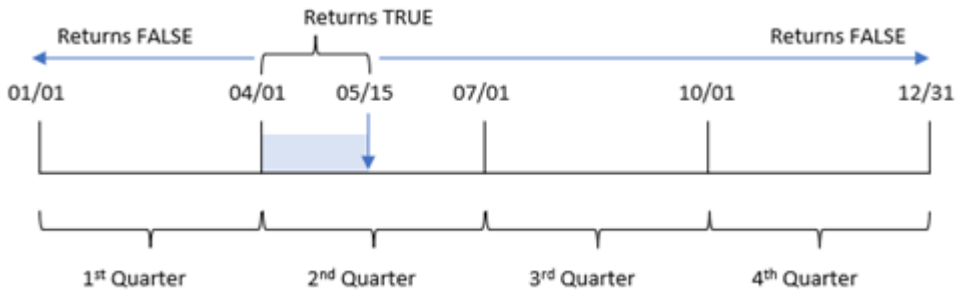
```
=inquartertodate(date,'05/15/2022', 0)
```

결과 테이블

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

in_quarter_to_date 측정값은 inquartertodate() 함수를 사용하여 차트 개체에 만들어집니다. 첫 번째 인수는 평가되는 날짜 필드입니다. 두 번째 인수는 5월 15일에 대한 하드 코딩된 날짜이며, 세그먼트로 분할할 분기를 식별하고 해당 세그먼트의 끝 경계를 정의하는 base_date입니다. period_no 0은 마지막 인수이며, 이는 함수가 세그먼트로 분할된 분기 앞이나 뒤에 있는 분기를 비교하지 않음을 의미합니다.

inquartertodate 함수 다이어그램, 차트 개체 예



4월 1일과 5월 15일 사이에 발생하는 모든 트랜잭션은 부울 결과 TRUE를 반환합니다. 5월 16일 이후의 트랜잭션은 4월 1일 이전의 트랜잭션과 마찬가지로 FALSE를 반환됩니다.

예 5 – 시나리오

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Products라는 테이블에 로드되는 데이터 집합.
- 제품 ID, 제조 날짜 및 원가에 관한 정보.

2022년 5월 15일, 제조 과정에서 일부 장비 오류가 발견되어 해결되었습니다. 이 날짜까지 해당 분기에 제조된 제품은 결함이 있습니다. 최종 사용자는 제품이 '결함' 또는 '무결함'인 상태와 해당 분기에서 현재까지 제조된 제품의 비용을 분기 이름별로 표시하는 차트 개체를 원합니다.

로드 스크립트

```
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
];
```

결과

다음과 같이 하십시오.

1. 데이터를 로드하고 시트를 엽니다. 새 테이블을 만듭니다. 분기 이름을 표시할 차원을 만듭니다.
   `=quartername(manufacture_date)`
2. 다음으로 결함이 있는 제품과 무결함인 제품을 식별하는 차원을 만듭니다.
   `=if(inquartertodate(manufacture_date,makedate(2022,05,15),0),'Defective','Faultless')`
3. 제품의 `cost_price` 합계를 계산하는 측정값을 만듭니다.
   `=sum(cost_price)`
4. 측정값의 **숫자 형식**을 **화폐**로 설정합니다.

결과 테이블

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

`inquartertodate()` 함수는 각 제품의 제조 날짜를 평가할 때 부울 값을 반환합니다. 부울 값 `TRUE`를 반환하는 경우 제품을 '`Defective`'로 표시합니다. `FALSE` 값을 반환하므로 5월 15일을 포함하는 분기까지 제조되지 않은 제품의 경우 제품을 '`Faultless`'로 표시합니다.

inweek

timestamp가 **base_date**를 포함하는 주에 속할 경우 이 함수는 True를 반환합니다.

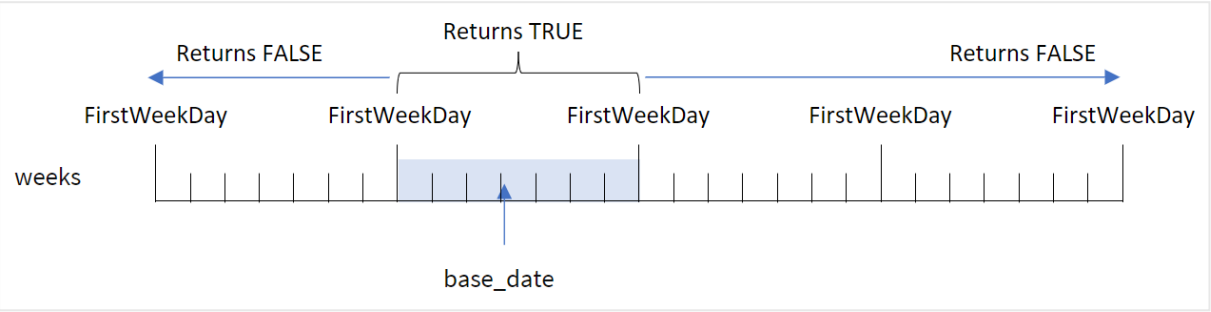
구문:

```
InWeek (timestamp, base_date, period_no[, first_week_day])
```

반환 데이터 유형: 부울

Qlik Sense에서 부울 true 값은 -1로 표시되고 false 값은 0으로 표시됩니다.

inweek() 함수의 범위 다이어그램



inweek() 함수는 base_date 인수를 사용하여 날짜가 속하는 7일 기간을 식별합니다. 주의 시작 요일은 FirstWeekDay 시스템 변수를 기반으로 합니다. 그러나 inweek() 함수의 first_week_day 인수를 사용하여 주의 첫 번째 요일을 변경할 수도 있습니다.

선택한 주가 정의된 후 함수는 규정된 날짜 값을 해당 주 세그먼트와 비교할 때 부울 결과를 반환합니다.

사용 시기

inweek() 함수는 부울 결과를 반환합니다. 일반적으로 이 유형의 함수는 if expression의 조건으로 사용됩니다. inweek() 함수는 평가된 날짜가 base_date 인수의 선택된 날짜가 있는 주에 발생했는지 여부에 따라 집계 또는 계산을 반환합니다.

예를 들어, inweek() 함수를 사용하여 특정 주에 제조된 모든 장비를 식별할 수 있습니다.

인수

인수	설명
timestamp	**base_date**와 비교할 날짜입니다.
base_date	주를 평가하는 데 사용되는 날짜입니다.
period_no	주는 **period_no**로 오프셋을 지정할 수 있습니다. **period_no**는 정수이며, 값 0은 **base_date**를 포함하는 주를 나타냅니다. **period_no**가 음수 값일 경우 이전 주, 양수 값일 경우 다음 주를 나타냅니다.
first_week_day	기본적으로 주의 첫 번째 요일은 토요일과 일요일 사이의 자정에 시작하는 일요일(FirstWeekDay 시스템 변수에 의해 확인됨)입니다. **first_week_day** 매개 변수가 **FirstWeekDay** 변수를 대체합니다. 다른 날에 시작하는 주를 나타내려면 0에서 6 사이의 플래그를 지정합니다.

first_week_day 값

일	Value
월요일	0
화요일	1
수요일	2
목요일	3
금요일	4
토요일	5
일요일	6

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 `SET DateFormat` 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

함수 예

예	결과
`inweek ('01/12/2006', '01/14/2006', 0)`	TRUE 반환
`inweek ('01/12/2006', '01/20/2006', 0 )`	FALSE 반환
`inweek ('01/12/2006', '01/14/2006', -1 )`	FALSE 반환
`inweek ('01/07/2006', '01/14/2006', -1)`	TRUE 반환
`inweek ('01/12/2006', '01/09/2006', 0, 3)`	`first_week_day`가 3(목요일)으로 지정되어 있으므로 FALSE를 반환합니다. 01/12/2006이 01/09/2006을 포함하는 주 다음 주의 첫 번째 날짜가 됩니다.

다음 항목은 이 함수를 사용하는 데 도움이 될 수 있습니다.

관련 항목

항목	기본 플래그/값	설명
FirstWeekDay (page 219)	6/Sunday	각 주의 시작 날짜를 정의합니다.

예 1 – 추가 인수 없음

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 'Transactions'라는 테이블에 로드되는 2022년 1월의 트랜잭션 집합이 포함된 데이터 집합.
- 6(일요일)으로 설정된 FirstWeekDay 시스템 변수.
- 다음을 포함하는 선행 LOAD:
  - inweek() 함수, 2022년 1월 14일 주에 발생한 트랜잭션을 확인하는 필드 'in_week'로 설정됩니다.
  - weekday() 함수, 각 날짜에 해당하는 요일을 표시하는 필드 'week_day'로 설정됩니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- week_day
- in_week

결과 테이블

date	week_day	in_week
01/02/2022	Sun	0
01/05/2022	Wed	0
01/06/2022	Thu	0
01/08/2022	Sat	0
01/09/2022	Sun	-1
01/10/2022	Mon	-1
01/11/2022	Tue	-1
01/12/2022	Wed	-1
01/13/2022	Thu	-1
01/14/2022	Fri	-1
01/15/2022	Sat	-1
01/16/2022	Sun	0
01/17/2022	Mon	0
01/18/2022	Tue	0
01/26/2022	Wed	0
01/27/2022	Thu	0
01/28/2022	Fri	0
01/29/2022	Sat	0
01/30/2022	Sun	0
01/31/2022	Mon	0

in_week 필드는 inweek() 함수를 사용하여 선행 LOAD 문에서 만들어집니다. 첫 번째 인수는 평가 중인 필드를 식별합니다. 두 번째 인수는 1월 14일에 대한 하드 코딩된 날짜이며 base_date입니다. base_date 인수는 FirstWeekDay 시스템 변수와 함께 작동하여 비교 대상 주를 식별합니다. period_no 0(함수가 분할된 주 앞이나 뒤에 있는 주를 비교하지 않음을 의미함)이 마지막 인수입니다.

FirstWeekDay 시스템 변수는 주가 일요일에 시작하여 토요일에 끝나도록 확인합니다. 따라서 1월은 아래 다이어그램에 따라 주로 나뉘며 1월 9일과 15일 사이의 날짜가 inweek() 계산에 유효한 기간을 제공합니다.

inweek() 함수 범위가 강조 표시된 캘린더의 다이어그램

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					

1월 9일과 1월 15일 사이에 발생하는 모든 트랜잭션은 부울 결과 TRUE를 반환합니다.

예 2 – period_no

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 'Transactions'라는 테이블에 로드되는 2022년 트랜잭션 집합을 포함하는 동일한 데이터 집합.
- 6(일요일)으로 설정된 FirstWeekDay 시스템 변수.
- 다음을 포함하는 선행 LOAD:
    - inweek () 함수, 2022년 1월 14일 주 이전의 전체 주에 발생하는 트랜잭션을 확인하는 필드 ''prev_week로 설정됩니다.
    - weekday() 함수, 각 날짜에 해당하는 요일을 표시하는 필드 'week_day'로 설정됩니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- week_day

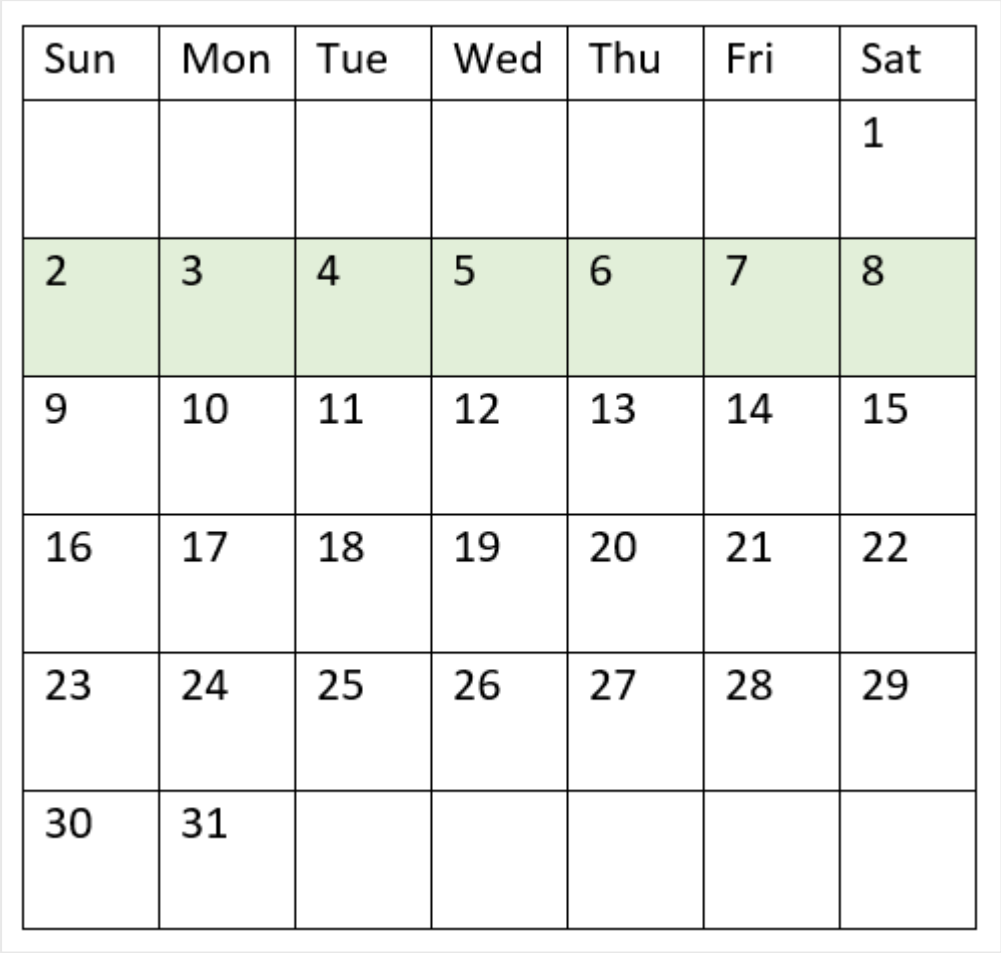
- prev_week

결과 테이블

date	week_day	prev_week
01/02/2022	Sun	-1
01/05/2022	Wed	-1
01/06/2022	Thu	-1
01/08/2022	Sat	-1
01/09/2022	Sun	0
01/10/2022	Mon	0
01/11/2022	Tue	0
01/12/2022	Wed	0
01/13/2022	Thu	0
01/14/2022	Fri	0
01/15/2022	Sat	0
01/16/2022	Sun	0
01/17/2022	Mon	0
01/18/2022	Tue	0
01/26/2022	Wed	0
01/27/2022	Thu	0
01/28/2022	Fri	0
01/29/2022	Sat	0
01/30/2022	Sun	0
01/31/2022	Mon	0

inweek() 함수에서 period_no 인수로 -1을 사용하면 비교 대상 주의 경계가 전체 7일만큼 뒤로 시프트합니다. period_no가 0이면 주는 1월 9일에서 15일 사이가 됩니다. 그러나 이 예에서 period_no가 -1이면 이 세그먼트의 시작과 끝 경계가 1주일 뒤로 시프트합니다. 날짜 경계는 1월 2일 ~ 1월 8일입니다.

inweek() 함수 범위가 강조 표시된 캘린더의 다이어그램



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					

따라서 1월 2일과 1월 8일 사이에 발생하는 모든 트랜잭션은 부울 결과 TRUE를 반환합니다.

예 3 – first_week_day

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 'Transactions'라는 테이블에 로드되는 2022년 트랜잭션 집합을 포함하는 동일한 데이터 집합.
- 6(일요일)으로 설정된 FirstWeekDay 시스템 변수.
- 다음을 포함하는 선행 LOAD:
  - inweek() 함수, 2022년 1월 14일 주에 발생한 트랜잭션을 확인하는 필드 'in_week'로 설정됩니다.
  - weekday() 함수, 각 날짜에 해당하는 요일을 표시하는 필드 'week_day'로 설정됩니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- week_day
- in_week

결과 테이블

date	week_day	in_week
01/02/2022	Sun	0
01/05/2022	Wed	0

date	week_day	in_week
01/06/2022	Thu	0
01/08/2022	Sat	0
01/09/2022	Sun	0
01/10/2022	Mon	-1
01/11/2022	Tue	-1
01/12/2022	Wed	-1
01/13/2022	Thu	-1
01/14/2022	Fri	-1
01/15/2022	Sat	-1
01/16/2022	Sun	-1
01/17/2022	Mon	0
01/18/2022	Tue	0
01/26/2022	Wed	0
01/27/2022	Thu	0
01/28/2022	Fri	0
01/29/2022	Sat	0
01/30/2022	Sun	0
01/31/2022	Mon	0

`inweek()` 함수의 `first_week_day` 인수로 `0`을 사용하면 `FirstWeekDay` 시스템 변수가 대체되고 월요일이 주의 첫 번째 요일로 설정됩니다.

inweek() 함수 범위가 강조 표시된 캘린더의 다이어그램

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					

따라서 1월 10일에서 16일 사이에 발생하는 모든 트랜잭션은 부울 결과 TRUE를 반환합니다.

예 4 - 차트 개체 예

로드 스크립트 및 차트 표현식

개요

첫 번째 예와 동일한 데이터 집합 및 시나리오가 사용됩니다.

그러나 이 예에서 데이터 집합은 변경되지 않고 응용 프로그램에 로드됩니다. 결과 테이블에서 측정값을 만들어 2022년 1월 14일 주에 발생한 트랜잭션을 확인합니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다.

- date

다음 측정값을 만듭니다.

- =inweek (date,'01/14/2022',0), 트랜잭션이 1월 14일과 같은 주에 발생했는지 여부를 계산합니다.
- =weekday(date), 각 날짜에 해당하는 요일을 표시합니다.

결과 테이블

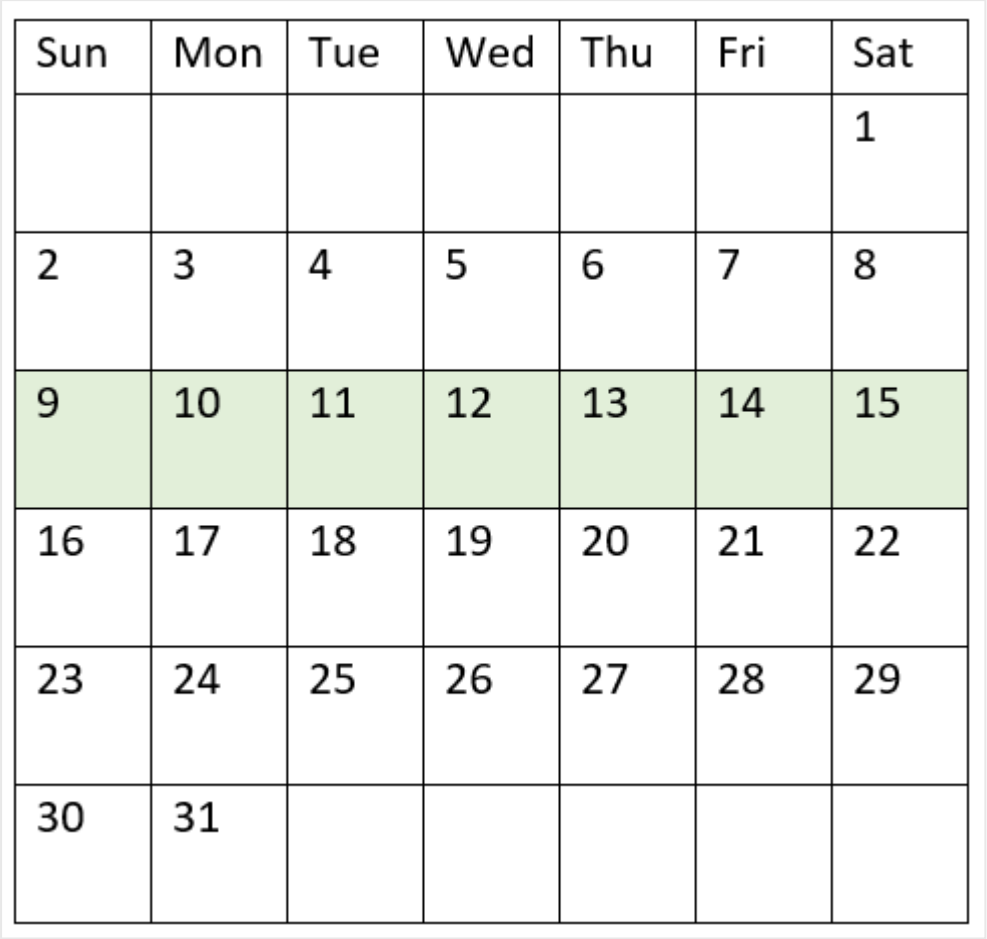
date	week_day	=inweek (date,'01/14/2022',0)
01/02/2022	Sun	0
01/05/2022	Wed	0
01/06/2022	Thu	0
01/08/2022	Sat	0
01/09/2022	Sun	-1
01/10/2022	Mon	-1
01/11/2022	Tue	-1
01/12/2022	Wed	-1

date	week_day	=inweek (date,'01/14/2022',0)
01/13/2022	Thu	-1
01/14/2022	Fri	-1
01/15/2022	Sat	-1
01/16/2022	Sun	0
01/17/2022	Mon	0
01/18/2022	Tue	0
01/26/2022	Wed	0
01/27/2022	Thu	0
01/28/2022	Fri	0
01/29/2022	Sat	0
01/30/2022	Sun	0
01/31/2022	Mon	0

'in_week' 측정값은 inweek() 함수를 사용하여 차트에 만들어집니다. 첫 번째 인수는 평가 중인 필드를 식별합니다. 두 번째 인수는 1월 14일에 대한 하드 코딩된 날짜이며 base_date입니다. base_date 인수는 FirstWeekDay 시스템 변수와 함께 작동하여 비교 대상 주를 식별합니다. period_no 0이 마지막 인수입니다.

FirstWeekDay 시스템 변수는 주가 일요일에 시작하여 토요일에 끝나도록 확인합니다. 따라서 1월은 아래 다이어그램에 따라 주로 나뉘며 1월 9일과 15일 사이의 날짜가 inweek() 계산에 유효한 기간을 제공합니다.

inweek() 함수 범위가 강조 표시된 캘린더의 다이어그램



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					

1월 9일과 1월 15일 사이에 발생하는 모든 트랜잭션은 부울 결과 TRUE를 반환합니다.

예 5 - 시나리오

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 'Products'라는 테이블에 로드되는 데이터 집합.
- 테이블에는 다음 필드가 포함됩니다.
  - 제품 ID
  - 제품 유형
  - 제조 날짜
  - 가격

장비 오류로 인해 1월 12일 주에 제조된 제품에 결함이 있는 것으로 확인되었습니다. 최종 사용자는 제조된 제품이 '결함' 또는 '무결함'인 상태와 해당 주에 제조된 제품의 비용을 주별로 표시하는 차트를 원합니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다.

- `=weekname(manufacture_date)`

다음 측정값을 만듭니다.

- `=if(only(inweek(manufacture_date,makedate(2022,01,12),0)),'Defective','Faultless')`, `inweek()` 함수를 사용하여 결함이 있는 제품과 결함이 없는 제품을 식별합니다.
- `=sum(cost_price)`, 각 제품의 비용 합계를 표시합니다.

다음과 같이 하십시오.

1. 측정값의 **숫자 형식**을 **화폐**로 설정합니다.
2. **모양**에서 **합계**를 끕니다.

결과 테이블

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

inweek() 함수는 각 제품의 제조 날짜를 평가할 때 부울 값을 반환합니다. 1월 12일이 속한 주에 제조된 모든 제품의 경우 inweek() 함수는 부울 값 TRUE를 반환하고 제품을 'Defective'로 표시합니다. 값 FALSE를 반환하여 해당 주에 제조되지 않은 제품의 경우 제품을 'Faultless'로 표시합니다.

inweektodate

이 함수는 **timestamp**가 **base_date**의 마지막 밀리초까지 포함하여 **base_date**를 포함한 주의 일부에 속할 경우 True를 반환합니다.

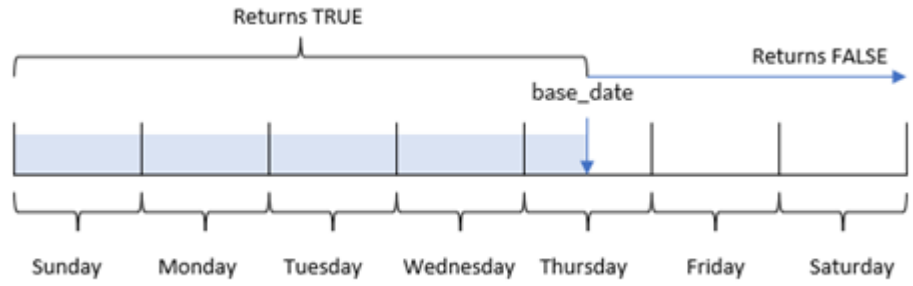
구문:

```
InWeekToDate (timestamp, base_date, period_no [, first_week_day])
```

반환 데이터 유형: 부울

> *Qlik Sense에서 부울 true 값은 -1로 표시되고 false 값은 0으로 표시됩니다.*

inweektodate 함수 다이어그램



inweektodate() 함수는 base_date 매개 변수를 사용하여 FirstWeekDay 시스템 변수(또는 사용자 정의 first_week_day 매개 변수)를 기반으로 하는 주 세그먼트의 최대 경계 날짜와 해당 주의 시작 날짜를 식별합니다. 이번 주 세그먼트가 정의되면 이 함수는 규정된 날짜 값을 해당 세그먼트와 비교할 때 부울 결과를 반환합니다.

사용 시기

inweektodate() 함수는 부울 결과를 반환합니다. 일반적으로 이 유형의 함수는 if 표현식의 조건으로 사용됩니다. 평가된 날짜가 특정 날짜를 포함하는 분기까지의 해당 주 동안 발생했는지 여부에 따라 집계 또는 계산을 반환합니다.

예를 들어, inweektodate() 함수를 사용하여 특정 날짜까지 지정된 주 동안 발생한 모든 판매를 계산할 수 있습니다.

인수

인수	설명
timestamp	**base_date**와 비교할 날짜입니다.
base_date	주를 평가하는 데 사용되는 날짜입니다.
period_no	주는 **period_no**로 오프셋을 지정할 수 있습니다. **period_no**는 정수이며, 값 0은 **base_date**를 포함하는 주를 나타냅니다. **period_no**가 음수 값일 경우 이전 주, 양수 값일 경우 다음 주를 나타냅니다.
first_week_day	기본적으로 주의 첫 번째 요일은 토요일과 일요일 사이의 자정에 시작하는 일요일입니다(FirstWeekDay 시스템 변수에 의해 결정됨). **first_week_day** 매개 변수가 **FirstWeekDay** 변수를 대체합니다. 다른 날에 시작하는 주를 나타내려면 0에서 6 사이의 플래그를 지정합니다. 월요일에 시작하여 일요일에 끝나는 주의 플래그는 월요일에 0, 화요일에 1, 수요일에 2, 목요일에 3, 금요일에 4, 토요일에 5, 일요일에 6을 사용합니다.

함수 예

예	상호 작용
inweektodate ('01/12/2006', '01/12/2006', 0)	TRUE을 반환합니다.
inweektodate ('01/12/2006', '01/11/2006', 0)	FALSE를 반환합니다.
inweektodate ('01/12/2006', '01/18/2006', -1)	FALSE을 반환합니다. period_no가 -1로 지정되어 있기 때문으로, timestamp가 측정되는 유효 데이터는 01/11/2006입니다.
inweektodate ('01/11/2006', '01/12/2006', 0, 3)	first_week_day가 3(목요일)으로 지정되어 있으므로 FALSE를 반환합니다. 01/12/2006이 01/12/2006를 포함하는 주의 다음 주 첫 번째 날짜가 됩니다.

다음 항목은 이 함수를 사용하는 데 도움이 될 수 있습니다.

관련 항목

항목	기본 플래그/값	설명
FirstWeekDay (page 219)	6/Sunday	각 주의 시작 날짜를 정의합니다.

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

예 1 – 추가 인수 없음

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 'Transactions'라는 테이블에 로드되는 2022년 1월의 트랜잭션 집합이 포함된 데이터 집합.
- TimestampFormat='M/D/YYYY h:mm:ss[.fff]' 서식으로 제공된 데이터 필드.
- 2022년 1월 14일까지의 주에 발생한 트랜잭션을 결정하는 필드 in_week_to_date 만들기.
- weekday() 함수를 사용하여 weekday라는 추가 필드 만들기. 이 새 필드는 각 날짜에 해당하는 요일을 표시하기 위해 만들어집니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- week_day
- in_week_to_date

결과 테이블

date	week_day	in_week_to_date
2022-01-02 12:22:06	Sun	0
2022-01-05 01:02:30	Wed	0
2022-01-06 15:36:20	Thu	0
2022-01-08 10:58:35	Sat	0
2022-01-09 08:53:32	Sun	-1
2022-01-10 21:13:01	Mon	-1
2022-01-11 00:57:13	Tue	-1
2022-01-12 09:26:02	Wed	-1
2022-01-13 15:05:09	Thu	-1
2022-01-14 18:44:57	Fri	-1
2022-01-15 06:10:46	Sat	0
2022-01-16 06:39:27	Sun	0

date	week_day	in_week_to_date
2022-01-17 10:44:16	Mon	0
2022-01-18 18:48:17	Tue	0
2022-01-26 04:36:03	Wed	0
2022-01-27 08:07:49	Thu	0
2022-01-28 12:24:29	Fri	0
2022-01-30 11:56:56	Sun	0
2022-01-30 14:40:19	Sun	0
2022-01-31 05:28:21	Mon	0

'in_week_to_date' 필드는 inweektodate() 함수를 사용하여 선행 LOAD 문에서 만들어집니다. 제공된 첫 번째 인수는 평가 중인 필드를 식별합니다. 두 번째 인수는 1월 14일에 대한 하드 코딩된 날짜이며, 세그먼트로 분할할 주를 식별하고 해당 세그먼트의 끝 경계를 정의하는 base_date입니다. period_no 0은 마지막 인수이며, 이는 함수가 세그먼트로 분할된 주 앞이나 뒤에 있는 주를 비교하지 않음을 의미합니다.

FirstWeekDay 시스템 변수는 주가 일요일에 시작하여 토요일에 끝나도록 결정합니다. 따라서 1월은 아래 다이어그램에 따라 주간으로 나뉘며 1월 9일과 14일 사이의 날짜가 inweekdodate() 계산에 유효한 기간을 제공합니다.

부울 결과 TRUE를 반환하는 트랜잭션 날짜를 보여 주는 캘린더 다이어그램

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					

1월 9일과 14일 사이에 발생하는 모든 트랜잭션은 부울 결과 TRUE를 반환합니다. 이 날짜 전후의 트랜잭션은 부울 결과 FALSE를 반환합니다.

예 2 – period_no

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 첫 번째 예와 동일한 데이터 집합 및 시나리오.
- 2022년 1월 14일에 끝나는 주 세그먼트 이전의 전체 주에 발생하는 트랜잭션을 결정하는 필드 prev_week_to_date 만들기.
- weekday() 함수를 사용하여 weekday라는 추가 필드 만들기. 이는 각 날짜에 해당하는 요일을 표시하기 위한 것입니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- week_day
- prev_week_to_date

결과 테이블

date	week_day	prev_week_to_date
2022-01-02 12:22:06	Sun	-1
2022-01-05 01:02:30	Wed	-1
2022-01-06 15:36:20	Thu	-1
2022-01-08 10:58:35	Sat	0
2022-01-09 08:53:32	Sun	0
2022-01-10 21:13:01	Mon	0
2022-01-11 00:57:13	Tue	0
2022-01-12 09:26:02	Wed	0
2022-01-13 15:05:09	Thu	0
2022-01-14 18:44:57	Fri	0
2022-01-15 06:10:46	Sat	0
2022-01-16 06:39:27	Sun	0
2022-01-17 10:44:16	Mon	0
2022-01-18 18:48:17	Tue	0
2022-01-26 04:36:03	Wed	0
2022-01-27 08:07:49	Thu	0
2022-01-28 12:24:29	Fri	0
2022-01-30 11:56:56	Sun	0
2022-01-30 14:40:19	Sun	0
2022-01-31 05:28:21	Mon	0

period_no 값 -1은 inweektodate () 함수가 입력 분기 세그먼트를 이전 주와 비교함을 나타냅니다. 주 세그먼트는 처음에 1월 9일과 1월 14일 사이에 해당합니다. 그런 다음 period_no는 이 세그먼트의 시작 및 끝 경계를 1주일 앞당겨 오프셋하여 날짜 경계가 1월 2일에서 1월 7일이 되도록 합니다.

부울 결과 TRUE를 반환하는 트랜잭션 날짜를 보여 주는 캘린더 다이어그램

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					

따라서 1월 2일과 8일 사이에 발생하는 모든 트랜잭션(1월 8일은 제외)은 부울 결과 TRUE를 반환합니다.

예 3 – first_week_day

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 첫 번째 예와 동일한 데이터 집합 및 시나리오.
- 2022년 1월 14일까지의 주에 발생한 트랜잭션을 결정하는 필드 in_week_to_date 만들기.
- weekday() 함수를 사용하여 weekday라는 추가 필드 만들기. 이는 각 날짜에 해당하는 요일을 표시하기 위한 것입니다.

이 예에서는 월요일을 주의 첫 번째 요일로 간주합니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- week_day
- in_week_to_date

결과 테이블

date	week_day	in_week_to_date
2022-01-02 12:22:06	Sun	0
2022-01-05 01:02:30	Wed	0
2022-01-06 15:36:20	Thu	0
2022-01-08 10:58:35	Sat	0
2022-01-09 08:53:32	Sun	0
2022-01-10 21:13:01	Mon	-1
2022-01-11 00:57:13	Tue	-1
2022-01-12 09:26:02	Wed	-1
2022-01-13 15:05:09	Thu	-1
2022-01-14 18:44:57	Fri	-1
2022-01-15 06:10:46	Sat	0

date	week_day	in_week_to_date
2022-01-16 06:39:27	Sun	0
2022-01-17 10:44:16	Mon	0
2022-01-18 18:48:17	Tue	0
2022-01-26 04:36:03	Wed	0
2022-01-27 08:07:49	Thu	0
2022-01-28 12:24:29	Fri	0
2022-01-30 11:56:56	Sun	0
2022-01-30 14:40:19	Sun	0
2022-01-31 05:28:21	Mon	0

inweektodate() 함수의 first_week_day 인수로 0을 사용하면 이 함수 인수가 FirstWeekDay 시스템 변수를 대체하고 월요일을 주의 첫 번째 요일로 설정합니다.

부울 결과 TRUE를 반환하는 트랜잭션 날짜를 보여 주는 캘린더 다이어그램

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						

따라서 1월 10일에서 14일 사이에 발생하는 모든 트랜잭션은 부울 결과 TRUE를 반환하는 반면 날짜가 이 경계를 벗어난 트랜잭션은 FALSE 값을 반환합니다.

예 4 – 차트 개체 예

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

이 로드 스크립트에는 첫 번째 예와 동일한 데이터 집합 및 시나리오가 포함되어 있습니다. 그러나 이 예에서는 변경되지 않은 데이터 집합이 응용 프로그램에 로드됩니다. 2022년 1월 14일까지의 주에 발생한 트랜잭션을 결정하는 계산이 차트 개체의 측정값으로 만들어집니다.

로드 스크립트

```
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
];
```

결과

다음과 같이 하십시오.

1. 데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다. `date`.
2. 트랜잭션이 1월 14일까지 같은 주에 발생했는지 여부를 계산하기 위해 다음 측정값을 만듭니다.
   `=inweektodate(date,'01/14/2022',0)`
3. 각 날짜에 해당하는 요일을 표시하려면 추가 측정값을 만듭니다.
   `=weekday(date)`

결과 테이블

date	week_day	in_week_to_date
2022-01-02 12:22:06	Sun	0

date	week_day	in_week_to_date
2022-01-05 01:02:30	Wed	0
2022-01-06 15:36:20	Thu	0
2022-01-08 10:58:35	Sat	0
2022-01-09 08:53:32	Sun	-1
2022-01-10 21:13:01	Mon	-1
2022-01-11 00:57:13	Tue	-1
2022-01-12 09:26:02	Wed	-1
2022-01-13 15:05:09	Thu	-1
2022-01-14 18:44:57	Fri	-1
2022-01-15 06:10:46	Sat	0
2022-01-16 06:39:27	Sun	0
2022-01-17 10:44:16	Mon	0
2022-01-18 18:48:17	Tue	0
2022-01-26 04:36:03	Wed	0
2022-01-27 08:07:49	Thu	0
2022-01-28 12:24:29	Fri	0
2022-01-30 11:56:56	Sun	0
2022-01-30 14:40:19	Sun	0
2022-01-31 05:28:21	Mon	0

in_week_to_date 필드는 inweektodate() 함수를 사용하여 차트 개체에서 측정값으로 만들어집니다. 제공된 첫 번째 인수는 평가 중인 필드를 식별합니다. 두 번째 인수는 1월 14일에 대한 하드 코딩된 날짜이며, 세그먼트로 분할할 주를 식별하고 해당 세그먼트의 끝 경계를 정의하는 base_date입니다. period_no 0은 마지막 인수이며, 이는 함수가 세그먼트로 분할된 주 앞이나 뒤에 있는 주를 비교하지 않음을 의미합니다.

FirstWeekDay 시스템 변수는 주가 일요일에 시작하여 토요일에 끝나도록 결정합니다. 따라서 1월은 아래 다이어그램에 따라 주간으로 나뉘며 1월 9일과 14일 사이의 날짜가 inweekdodate() 계산에 유효한 기간을 제공합니다.

부울 결과 TRUE를 반환하는 트랜잭션 날짜를 보여 주는 캘린더 다이어그램

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					

1월 9일과 14일 사이에 발생하는 모든 트랜잭션은 부울 결과 TRUE를 반환합니다. 이 날짜 전후의 트랜잭션은 부울 결과 FALSE를 반환합니다.

예 5 – 시나리오

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Products라는 테이블에 로드되는 데이터 집합.
- 제품 ID, 제조 날짜 및 원가에 관한 정보.

장비 오류로 인해 1월 12일 주에 제조된 제품에 결함이 있는 것으로 확인되었습니다. 문제는 1월 13일에 해결되었습니다. 최종 사용자는 제조된 제품이 '결함' 또는 '무결함'인 상태와 해당 주에 제조된 제품의 비용을 주별로 표시하는 차트 개체를 원합니다.

로드 스크립트

```
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
];
```

결과

다음과 같이 하십시오.

1. 데이터를 로드하고 시트를 엽니다. 새 테이블을 만듭니다. 주 이름을 표시할 차원을 만듭니다.
   `=weekname(manufacture_date)`
2. 다음으로 결함이 있는 제품과 무결함인 제품을 식별하는 차원을 만듭니다.
   `=if(inWeektodate(manufacture_date,makedate(2022,01,12),0),'Defective','Faultless')`
3. 제품의 `cost_price` 합계를 계산하는 측정값을 만듭니다.
   `=sum(cost_price)`
4. 측정값의 **숫자 형식**을 **화폐**로 설정합니다.

결과 테이블

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

`inweektodate()` 함수는 각 제품의 제조 날짜를 평가할 때 부울 값을 반환합니다. 부울 값 `TRUE`를 반환하는 경우 제품을 `'Defective'`로 표시합니다. `FALSE` 값을 반환하는 제품의 경우 1월 12일까지의 주에 제조되지 않은 경우 제품을 `'Faultless'`로 표시합니다.

inyear

timestamp가 **base_date**를 포함하는 연도에 속할 경우 이 함수는 True를 반환합니다.

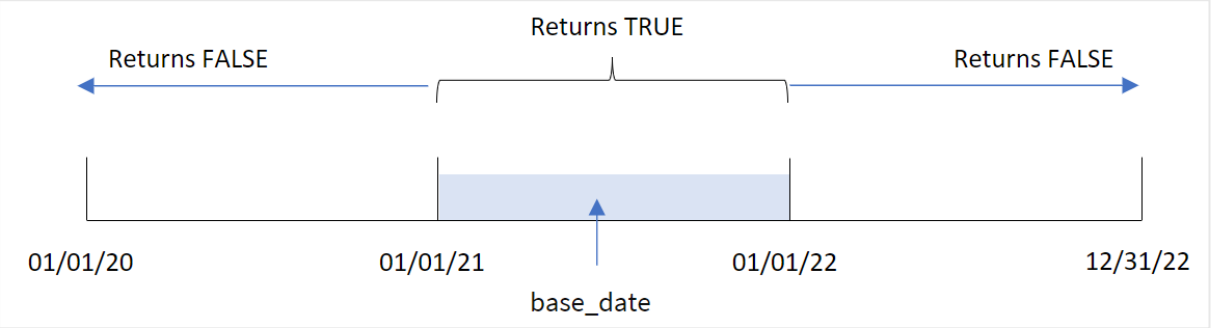
구문:

```
InYear (timestamp, base_date, period_no [, first_month_of_year])
```

반환 데이터 유형: 부울

Qlik Sense에서 부울 true 값은 -1로 표시되고 false 값은 0으로 표시됩니다.

inyear() 함수의 범위 다이어그램



inyear() 함수는 선택한 날짜 값을 base_date에서 정의한 연도와 비교할 때 부울 결과를 반환합니다.

사용 시기

inyear() 함수는 부울 결과를 반환합니다. 일반적으로 이 유형의 함수는 if expression의 조건으로 사용됩니다. 이는 평가된 날짜가 해당 연도에 발생했는지 여부에 따라 집계 또는 계산을 반환합니다. 예를 들어, inyear() 함수는 정의된 연도에 발생한 모든 판매를 식별하는 데 사용할 수 있습니다.

인수

인수	설명
timestamp	**base_date**와 비교할 날짜입니다.
base_date	연도를 평가하는 데 사용되는 날짜입니다.
period_no	연도는 **period_no**로 오프셋을 지정할 수 있습니다. **period_no**는 정수이며, 값 0은 **base_date**를 포함하는 연도를 나타냅니다. **period_no**가 음수 값일 경우 이전 연도, 양수 값일 경우 다음 연도를 나타냅니다.
first_month_of_year	1월에 시작되지 않는 (회계)연도를 사용하려는 경우 **first_month_of_year**에 2와 12 사이의 값을 지정하십시오.

다음 값을 사용하여 first_month_of_year 인수에서 연도의 첫 번째 달을 설정할 수 있습니다.

first_month_of_year 값

Month	Value
2월	2
3월	3
4월	4
5월	5

Month	Value
6월	6
7월	7
8월	8
9월	9
10월	10
11월	11
12월	12

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

함수 예

예	결과
`inyear ('01/25/2013', '01/01/2013', 0 )`	TRUE 반환
`inyear ('01/25/2012', '01/01/2013', 0)`	FALSE 반환
`inyear ('01/25/2013', '01/01/2013', -1)`	FALSE 반환
`inyear ('01/25/2012', '01/01/2013', -1 )`	TRUE 반환
`inyear ('01/25/2013', '01/01/2013', 0, 3)`	TRUE 반환 base_date 및 first_month_of_year 값은 타임스탬프가 01/03/2012와 02/28/2013 내에 있어야 함을 지정합니다.
`inyear ('03/25/2013', '07/01/2013', 0, 3 )`	TRUE 반환

예 1 - 기본 예

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 'Transactions'라는 테이블에 로드되는 2020년과 2022년 사이의 트랜잭션 집합이 포함된 데이터 집합.
- 'in_year' 필드로 설정된 inyear() 함수를 포함하고 2021년 7월 26일과 같은 연도에 발생한 트랜잭션을 확인하는 선행 LOAD.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

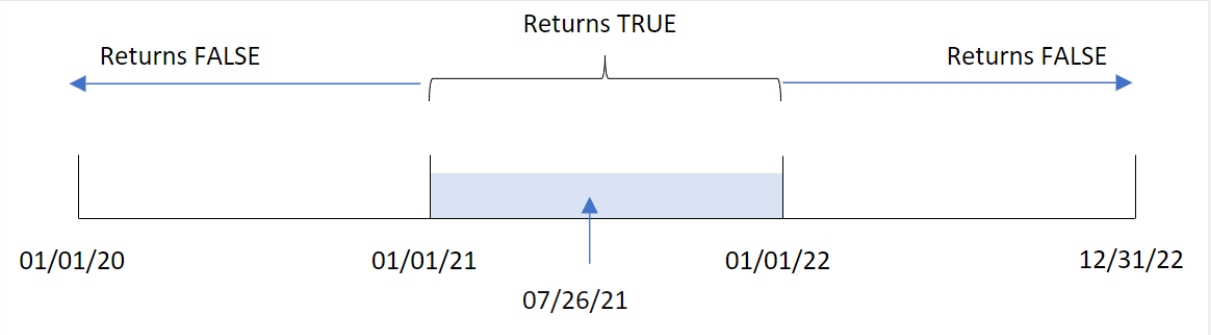
- date
- in_year

결과 테이블

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

in_year 필드는 inyear() 함수를 사용하여 선행 LOAD 문에서 만들어집니다. 첫 번째 인수는 평가 중인 필드를 식별합니다. 두 번째 인수는 2021년 7월 26일에 대한 하드 코딩된 날짜이며, 비교 대상 연도를 확인하는 base_date입니다. period_no 0은 마지막 인수이며, 이는 inyear() 함수가 연도 앞이나 뒤에 있는 연도를 비교하지 않음을 의미합니다.

7월 26일을 기준 날짜로 하는 inyear() 함수 범위의 다이어그램



2021년에 발생하는 모든 트랜잭션은 부울 결과 TRUE를 반환합니다.

예 2 – period_no

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 'Transactions'라는 테이블에 로드되는 2020년과 2022년 사이의 트랜잭션 집합이 포함된 데이터 집합.
- 'previous_year' 필드로 설정된 inyear() 함수를 포함하고 2021년 7월 26일을 포함하는 연도의 이전 연도에 발생한 트랜잭션을 확인하는 선행 LOAD.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- previous_year

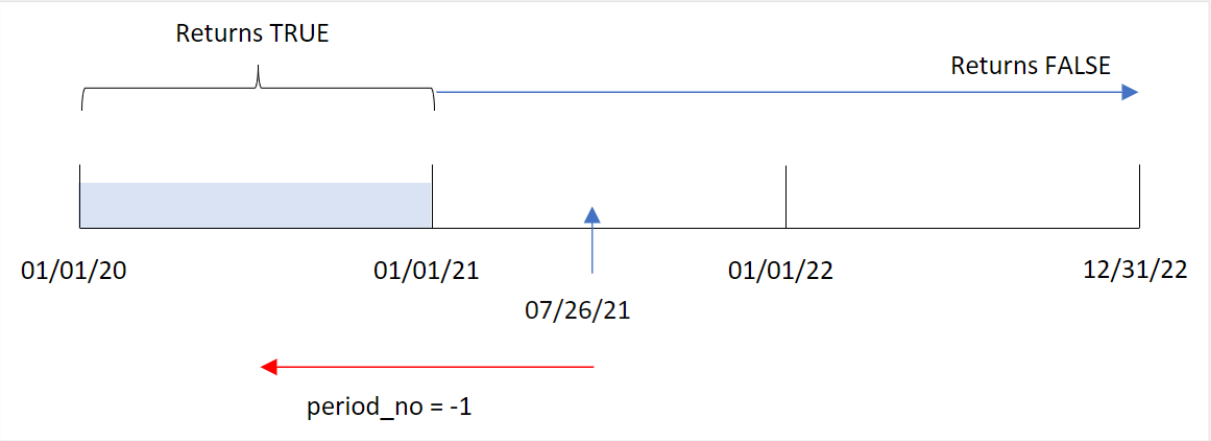
결과 테이블

date	previous_year
01/13/2020	-1
02/26/2020	-1
03/27/2020	-1
04/16/2020	-1
05/21/2020	-1
08/14/2020	-1
10/07/2020	-1
12/05/2020	-1
01/22/2021	0
02/03/2021	0
03/17/2021	0
04/23/2021	0
05/04/2021	0
06/30/2021	0
07/26/2021	0
12/27/2021	0
06/06/2022	0
07/18/2022	0

date	previous_year
11/14/2022	0
12/12/2022	0

inyear() 함수의 period_no 인수로 -1을 사용하면 비교 대상 연도의 경계가 1년 뒤로 시프트됩니다. 2021년이 초기에 비교 대상 연도로 식별됩니다. period_no는 비교 대상 연도를 1년 오프셋하여 2020년을 비교 대상 연도로 만듭니다.

period_no 인수가 -1로 설정된 inyear() 함수 범위의 다이어그램



따라서 2020년에 발생하는 모든 트랜잭션은 부울 결과 TRUE를 반환합니다.

예 3 - first_month_of_year

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 'Transactions'라는 테이블에 로드되는 2020년과 2022년 사이의 트랜잭션 집합이 포함된 데이터 집합.
- 'in_year' 필드로 설정된 inyear() 함수를 포함하고 2021년 7월 26일과 같은 연도에 발생한 트랜잭션을 확인하는 선행 LOAD.

그러나 이 예에서 조직 정책은 회계 연도의 첫 번째 달이 되는 3월입니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- in_year

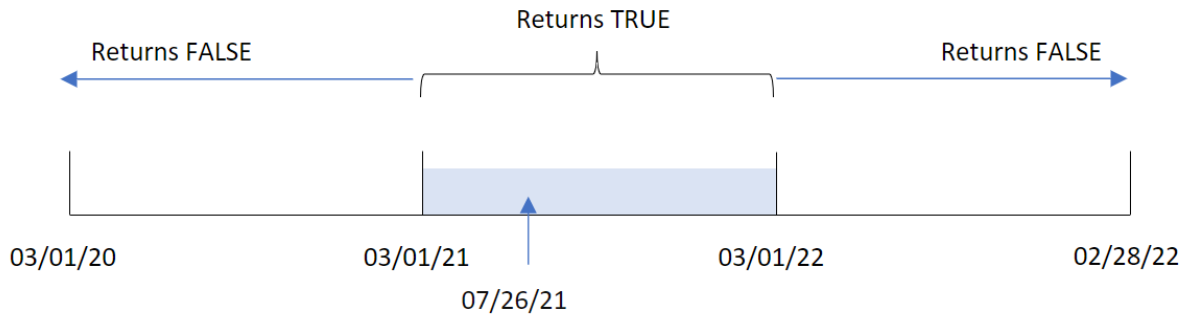
결과 테이블

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

date	in_year
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

inyear() 함수의 first_month_of_year 인수로 3을 사용하면 연도가 3월 1일에 시작되고 2월 말에 끝납니다.

3월을 연도의 첫 번째 달로 설정한 inyear() 함수 범위의 다이어그램



따라서 2021년 3월 1일과 2022년 3월 1일 사이에 발생하는 모든 트랜잭션은 부울 결과 TRUE를 반환합니다.

예 4 - 차트 개체 예

로드 스크립트 및 차트 표현식

개요

첫 번째 예와 동일한 데이터 집합 및 시나리오가 사용됩니다.

그러나 이 예에서 데이터 집합은 변경되지 않고 응용 프로그램에 로드됩니다. 트랜잭션이 2021년 7월 26일과 같은 연도에 발생했는지 여부를 확인하는 계산은 응용 프로그램의 차트 개체에서 측정값으로 만들어집니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다.

- date

트랜잭션이 2021년 7월 26일과 같은 연도에 발생했는지 여부를 계산하려면 다음 측정값을 만듭니다.

- =inyear(date,'07/26/2021', 0)

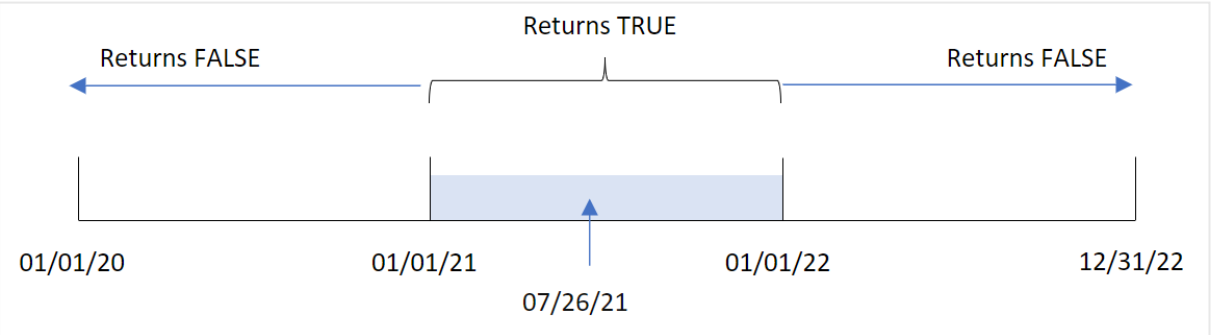
결과 테이블

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

date	=inyear(date,'07/26/2021',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

'in_year' 필드는 inyear() 함수를 사용하여 차트에 만들어집니다. 첫 번째 인수는 평가 중인 필드를 식별합니다. 두 번째 인수는 2021년 7월 26일에 대한 하드 코딩된 날짜이며, 비교 대상 연도를 확인하는 base_date입니다. period_no 0은 마지막 인수이며, 이는 inyear() 함수가 연도 앞이나 뒤에 있는 연도를 비교하지 않음을 의미합니다.

7월 27일을 기준 날짜로 하는 inyear() 함수 범위의 다이어그램



2021년에 발생하는 모든 트랜잭션은 부울 결과 TRUE를 반환합니다.

예 5 - 시나리오

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 'Products'라는 테이블에 로드되는 데이터 집합.
- 테이블에는 다음 필드가 포함됩니다.
    - 제품 ID
    - 제품 유형
    - 제조 날짜
    - 가격

최종 사용자는 2021년에 제조된 제품의 비용을 제품 유형별로 표시하는 차트 개체를 원합니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다.

- product_type

2021년에 제조된 각 제품의 합계를 계산하려면 다음 측정값을 만듭니다.

- =sum(if(InYear(manufacture_date,makedate(2021,01,01),0),cost_price,0))

다음과 같이 하십시오.

1. 측정값의 **숫자 형식**을 **화폐**로 설정합니다.
2. **모양**에서 **합계**를 끕니다.

결과 테이블

product_type	=sum(if(InYear(manufacture_date,makedate(2021,01,01),0),cost_price,0))
제품 A	$95.93
제품 B	$128.66
제품 C	$61.89
제품 D	$171.21

inyear() 함수는 각 제품의 제조 날짜를 평가할 때 부울 값을 반환합니다. 2021년에 제조된 모든 제품의 경우 inyear() 함수는 부울 값 TRUE를 반환하고 cost_price의 합계를 표시합니다.

inyeartodate

이 함수는 **timestamp**가 **base_date**의 마지막 밀리초까지 포함하여 **base_date**를 포함한 연도의 일부에 속할 경우 True를 반환합니다.

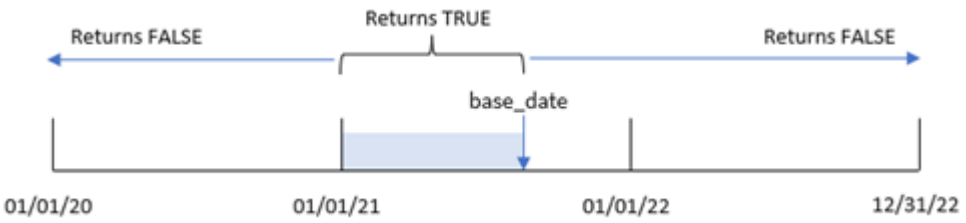
구문:

```
InYearToDate (timestamp, base_date, period_no[, first_month_of_year])
```

반환 데이터 유형: 부울

Qlik Sense에서 부울 true 값은 -1로 표시되고 false 값은 0으로 표시됩니다.

inyeartodate 함수의 다이어그램



inyeartodate() 함수는 base_date를 사용하여 연도의 특정 부분을 세그먼트로 분할하며, 해당 연도 세그먼트에 대해 허용되는 최대 날짜를 식별합니다. 그런 다음 이 함수는 날짜 필드 또는 값이 이 세그먼트에 속하는지 평가하고 부울 결과를 반환합니다.

인수

인수	설명
timestamp	**base_date**와 비교할 날짜입니다.
base_date	연도를 평가하는 데 사용되는 날짜입니다.
period_no	연도는 **period_no**로 오프셋을 지정할 수 있습니다. **period_no**는 정수이며, 값 0은 **base_date**를 포함하는 연도를 나타냅니다. **period_no**가 음수 값일 경우 이전 연도, 양수 값일 경우 다음 연도를 나타냅니다.
first_month_of_year	1월에 시작되지 않는 (회계)연도를 사용하려는 경우 **first_month_of_year**에 2와 12 사이의 값을 지정하십시오.

사용 시기

inyeartodate() 함수는 부울 결과를 반환합니다. 일반적으로 이 유형의 함수는 if 표현식의 조건으로 사용됩니다. 이는 평가된 날짜가 해당 날짜를 포함하여 해당 연도에 발생했는지 여부에 따라 집계 또는 계산을 반환합니다.

예를 들어, inyeartodate() 함수를 사용하여 특정 날짜까지 연도에 제조된 모든 장비를 식별할 수 있습니다.

이 예에서는 날짜 서식 MM/DD/YYYY를 사용합니다. 날짜 서식은 데이터 로드 스크립트 맨 위에서 SET DateFormat 문으로 지정됩니다. 이 예의 형식을 필요에 따라 변경하십시오.

함수 예

예	결과
inyeartodate ('01/25/2013', '02/01/2013', 0)	TRUE를 반환합니다.
inyeartodate ('01/25/2012', '01/01/2013', 0)	FALSE를 반환합니다.
inyeartodate ('01/25/2012', '02/01/2013', -1)	TRUE를 반환합니다.

예	결과
inyeartodate ('11/25/2012', '01/31/2013', 0, 4)	TRUE을 반환합니다. timestamp의 값이 네 번째 월에 시작되는 회계년도 내에 속하고 base_date 값 이전입니다.
inyeartodate ('3/31/2013', '01/31/2013', 0, 4)	FALSE을 반환합니다. 이전 예와 비교하여, timestamp의 값은 여전히 회계년도 내에 포함되지만 base_date 값 이후이므로 해당 연도의 일부를 벗어납니다.

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

예 1 – 추가 인수 없음

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Transactions라는 테이블에 로드되는 2020년과 2022년 사이의 트랜잭션 집합이 포함된 데이터 집합.
- DateFormat 시스템 변수(MM/DD/YYYY) 서식으로 제공된 날짜 필드.
- 2021년 7월 26일까지의 해당 연도에 발생한 트랜잭션을 결정하는 필드 in_year_to_date 만들기.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- `date`
- `in_year_to_date`

결과 테이블

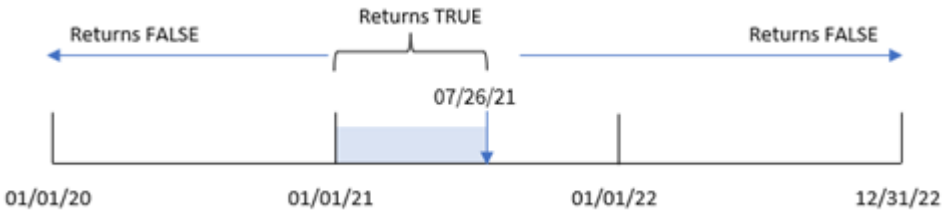
date	in_year_to_date
01/13/2020	0
02/26/2020	0
03/27/2020	0
04/16/2020	0
05/21/2020	0
06/14/2020	0
08/07/2020	0
09/05/2020	0
01/22/2021	-1
02/03/2021	-1
03/17/2021	-1

date	in_year_to_date
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

'in_year_to_date' 필드는 inyeartodate() 함수를 사용하여 선행 LOAD 문에서 만들어집니다. 제공된 첫 번째 인수는 평가 중인 필드를 식별합니다.

두 번째 인수는 2021년 7월 26일에 대한 하드 코딩된 날짜이며, 연도 세그먼트의 끝 경계를 식별하는 base_date입니다. period_no 0은 마지막 인수이며, 이는 함수가 세그먼트로 분할된 연도 앞이나 뒤에 있는 연도를 비교하지 않음을 의미합니다.

inyeartodate 함수의 다이어그램, 추가 인수 없음



1월 1일과 7월 26일 사이에 발생하는 모든 트랜잭션은 부울 결과 TRUE를 반환합니다. 2021년 이전 및 2021년 7월 26일 이후의 트랜잭션 날짜는 FALSE를 반환합니다.

예 2 – period_no

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 첫 번째 예와 동일한 데이터 집합 및 시나리오.
- 2021년 7월 26일에 끝나는 연도 세그먼트 이전의 전체 연도에 발생하는 트랜잭션을 결정하는 필드 previous_year_to_date 만들기.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- previous_year_to_date

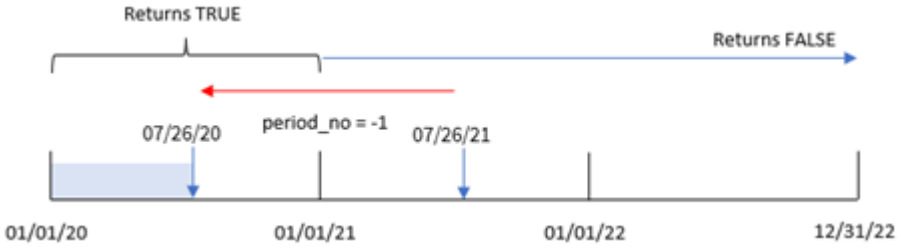
결과 테이블

date	previous_year_to_date
01/13/2020	-1
02/26/2020	-1
03/27/2020	-1
04/16/2020	-1

date	previous_year_to_date
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

period_no 값 -1은 inyeartodate () 함수가 입력 분기 세그먼트를 이전 연도와 비교함을 나타냅니다. 입력 날짜가 2021년 7월 26일인 경우 2021년 1월 1일부터 2021년 7월 26일까지의 세그먼트는 처음에 연간 누계로 식별되었습니다. 그런 다음 period_no는 이 세그먼트를 1년 앞서 오프셋하여 날짜 경계가 2020년 1월 1일에서 7월 26일이 되도록 합니다.

inyeartodate 함수의 다이어그램, period_no 예



따라서 2020년 1월 1일과 7월 26일 사이에 발생하는 모든 트랜잭션은 부울 결과 TRUE를 반환합니다.

예 3 – first_month_of_year

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 첫 번째 예와 동일한 데이터 집합 및 시나리오.
- 2021년 7월 26일까지의 같은 연도에 발생한 트랜잭션을 결정하는 필드 in_year_to_date 만들기.

이 예에서는 3월을 회계 연도의 첫 번째 달로 설정합니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

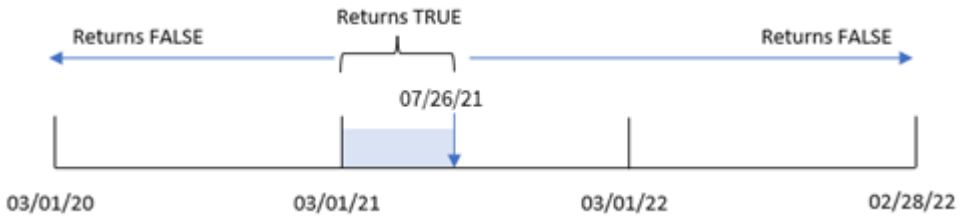
- date
- in_year_to_date

결과 테이블

date	in_year_to_date
01/13/2020	0
02/26/2020	0
03/27/2020	0
04/16/2020	0
05/21/2020	0
06/14/2020	0
08/07/2020	0
09/05/2020	0
01/22/2021	0
02/03/2021	0
03/17/2021	-1
04/23/2021	-1
05/04/2021	-1
06/30/2021	-1
07/26/2021	-1
07/27/2021	0
06/06/2022	0
07/18/2022	0
11/14/2022	0
12/12/2022	0

inyeartodate() 함수에서 3을 first_month_of_year 인수로 사용하면 이 함수는 3월 1일에 연도를 시작합니다. 그런 다음 2021년 7월 26일인 base_date가 해당 연도 세그먼트의 종료 날짜로 설정됩니다.

inyeartodate 함수의 다이어그램, first_month_of_year 예



따라서 2021년 3월 1일에서 7월 26일 사이에 발생하는 모든 트랜잭션은 부울 결과 TRUE를 반환하는 반면 날짜가 이 경계를 벗어난 트랜잭션은 FALSE의 값을 반환합니다.

예 4 – 차트 개체 예

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

이 로드 스크립트에는 첫 번째 예와 동일한 데이터 집합 및 시나리오가 포함되어 있습니다. 그러나 이 예에서는 변경되지 않은 데이터 집합이 응용 프로그램에 로드됩니다. 2021년 7월 26일까지 같은 연도에 발생한 트랜잭션을 확인하는 계산은 응용 프로그램의 차트 개체에서 측정값으로 만들어집니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다.date.

다음 측정값을 만듭니다.

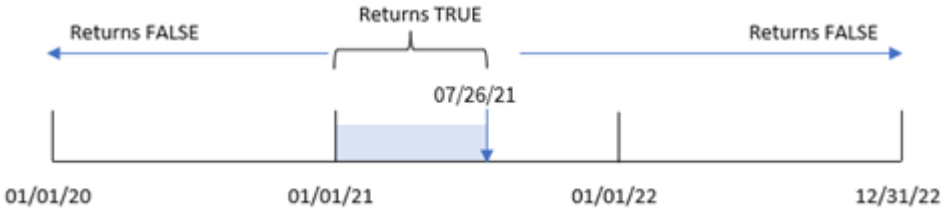
```
=inyeartodate(date,'07/26/2021', 0)
```

결과 테이블

date	=inyeartodate(date,'07/26/2021', 0)
01/13/2020	0
02/26/2020	0
03/27/2020	0
04/16/2020	0
05/21/2020	0
06/14/2020	0
08/07/2020	0
09/05/2020	0
01/22/2021	-1
02/03/2021	-1
03/17/2021	-1
04/23/2021	-1
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

in_year_to_date 측정값은 inyeartodate() 함수를 사용하여 차트 개체에 만들어집니다. 제공된 첫 번째 인수는 평가 중인 필드를 식별합니다. 두 번째 인수는 2021년 7월 26일에 대한 하드 코딩된 날짜이며, 비교 연도 세그먼트의 끝 경계를 식별하는 base_date입니다. period_no 0은 마지막 인수이며, 이는 함수가 세그먼트로 분할된 연도 앞이나 뒤에 있는 연도를 비교하지 않음을 의미합니다.

inyeartodate 함수 다이어그램, 차트 개체 예



2021년 1월 1일과 7월 26일 사이에 발생한 모든 트랜잭션은 부울 결과 TRUE를 반환합니다. 2021년 이전 및 2021년 7월 26일 이후의 트랜잭션 날짜는 FALSE를 반환합니다.

예 5 – 시나리오

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Products라는 테이블에 로드되는 데이터 집합.
- 제품 ID, 제품 유형, 제조 날짜 및 원가에 관한 정보.

최종 사용자는 2021년 7월 26일까지 제조된 제품의 원가를 제품 유형별로 표시하는 차트 개체를 원합니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다.product_type.

2021년 7월 27일 이전에 제조된 각 제품의 합계를 계산하는 측정값을 만듭니다.

```
=sum(if(inyeartodate(manufacture_date,makedate(2021,07,26),0),cost_price,0))
```

측정값의 **숫자 형식**을 **화폐**로 설정합니다.

결과 테이블

product_type	**=sum(if(inyeartodate(manufacture_date,makedate (2021,07,26),0),cost_price,0))**
제품 A	$95.93
제품 B	$128.66
제품 C	$61.89
제품 D	$146.09

inyeartodate() 함수는 각 제품의 제조 날짜를 평가할 때 부울 값을 반환합니다. 2021년 7월 27일 이전에 제조된 모든 제품의 경우 inyeartodate() 함수는 부울 값 TRUE를 반환하고 cost_price의 합계를 계산합니다.

제품 D는 2021년 7월 26일 이후에도 제조된 유일한 제품입니다. product_ID가 8203인 항목은 12월 27일에 제조되었으며 가격은 $25.12입니다. 따라서 이 비용은 차트 개체의 제품 D에 대한 합계에 포함되지 않았습니다.

lastworkdate

lastworkdate 함수는 선택적으로 나열된 **holiday**를 고려하여 **start_date**에 시작하는 경우 **no_of_workdays**(월요일-금요일)가 끝나는 가장 빠른 끝 날짜를 반환합니다. **start_date** 및 **holiday**는 유효한 날짜 또는 타임스탬프여야 합니다.

구문:

```
lastworkdate(start_date, no_of_workdays {, holiday})
```

반환 데이터 유형: 정수

lastworkdate() 함수가 어떻게 사용되는지 보여 주는 캘린더

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			

제한 사항

월요일에 시작하여 금요일에 끝나는 근무 주 이외의 다른 주와 관련된 지역 또는 시나리오에 대한 lastworkdate() 함수를 수정할 수 있는 방법은 없습니다.

holiday 매개 변수는 문자열 상수여야 합니다. 표현식을 허용하지 않습니다.

사용 시기

lastworkdate() 함수는 일반적으로 사용자가 프로젝트 시작 날짜와 해당 기간에 발생할 휴일을 기반으로 프로젝트의 제안된 종료 날짜 또는 할당을 계산하려고 할 때 표현식의 일부로 사용됩니다.

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

인수

인수	설명
start_date	평가할 시작 날짜입니다.
no_of_ workdays	근무해야 할 일수입니다.
holiday	근무일에서 제외시킬 공휴일 기간입니다. 휴일은 문자열 상수 날짜로 표시됩니다. 여러 휴일 날짜를 쉼표로 구분하여 지정할 수 있습니다. **'12/25/2013', '12/26/2013', '12/31/2013', '01/01/2014'**

예 1 - 기본 예

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 프로젝트 ID, 프로젝트 시작 날짜 및 프로젝트에 필요한 예상 공수(일)가 포함된 데이터 집합. 이 데이터 집합은 'Projects'라는 테이블에 로드됩니다.
- 필드 'end_date'로 설정되고 각 프로젝트의 예약된 종료 날짜를 식별하는 lastworkdate() 함수를 포함하는 선행 LOAD.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- id
- start_date
- effort
- end_date

결과 테이블

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

예정된 휴일이 없기 때문에 이 함수는 월요일부터 금요일까지 정의된 근무일 수를 시작 날짜에 추가하여 가능한 가장 빠른 종료 날짜를 찾습니다.

다음 캘린더에 작업일이 녹색으로 강조 표시된 프로젝트 3의 시작 날짜와 종료 날짜가 표시됩니다.

프로젝트 3의 시작 날짜와 종료 날짜를 표시하는 캘린더

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				

예 2 - 단일 휴일

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 프로젝트 ID, 프로젝트 시작 날짜 및 프로젝트에 필요한 예상 공수(일)가 포함된 데이터 집합. 이 데이터 집합은 'Projects'라는 테이블에 로드됩니다.
- 필드 'end_date'로 설정되고 각 프로젝트의 예약된 종료 날짜를 식별하는 lastworkdate() 함수를 포함하는 선행 LOAD.

단, 2022년 5월 18일에는 1일의 휴일이 예정되어 있습니다. 선행 LOAD에서 lastworkdate() 함수는 세 번째 인수에 휴일을 포함하여 각 프로젝트의 예약된 종료 날짜를 식별합니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- id
- start_date
- effort
- end_date

결과 테이블

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

lastworkdate() 함수의 세 번째 인수로 예정된 단일 휴일이 입력됩니다. 결과적으로 휴일이 종료 날짜 이전의 근무일 중에 발생하므로 프로젝트 3의 종료 날짜가 1일 뒤로 시프트됩니다.

다음 캘린더는 프로젝트 3의 시작 날짜와 종료 날짜를 보여 주고 휴일이 프로젝트의 종료 날짜가 1일 변경됨을 보여 줍니다.

5월 18일을 휴일로 하여 프로젝트 3의 시작 날짜와 종료 날짜를 표시하는 캘린더

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				

예 3 - 여러 휴일

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 프로젝트 ID, 프로젝트 시작 날짜 및 프로젝트에 필요한 예상 공수(일)가 포함된 데이터 집합. 이 데이터 집합은 'Projects'라는 테이블에 로드됩니다.
- 필드 'end_date'로 설정되고 각 프로젝트의 예약된 종료 날짜를 식별하는 lastworkdate() 함수를 포함하는 선행 LOAD.

단, 5월 19일, 20일, 21일, 22일 3개의 휴일이 있습니다. 선행 LOAD에서 lastworkdate() 함수는 세 번째 인수에 각 휴일을 포함하여 각 프로젝트의 예약된 종료 날짜를 식별합니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- id
- start_date
- effort
- end_date

결과 테이블

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

시작 날짜와 근무일 수 이후에 lastworkdate() 함수에 인수 목록으로 4개의 휴일이 입력됩니다.

다음 캘린더는 프로젝트 3의 시작 날짜와 종료 날짜를 보여 주고 휴일이 프로젝트의 종료 날짜가 3일 변경됨을 보여 줍니다.

5월 19일 ~ 22일을 휴일로 하여 프로젝트 3의 시작 날짜와 종료 날짜를 표시하는 캘린더

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				

예 4 - 단일 휴일(차트)

로드 스크립트 및 차트 표현식

개요

첫 번째 예와 동일한 데이터 집합 및 시나리오가 사용됩니다.

그러나 이 예에서 데이터 집합은 변경되지 않고 앱에 로드됩니다. end_date 필드는 차트에서 측정값으로 계산됩니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- id
- start_date
- effort

end_date를 계산하려면 다음 측정값을 만듭니다.

- =LastWorkDate(start_date,effort,'05/18/2022')

결과 테이블

id	start_date	effort	=LastWorkDate(start_date,effort,'05/18/2022')
1	01/01/2022	14	01/20/2022
2	02/10/2022	17	03/04/2022
3	05/17/2022	5	05/23/2022
4	06/01/2022	12	06/16/2022
5	08/10/2022	26	09/14/2022

예정된 단일 휴일은 차트에 측정값으로 입력됩니다. 결과적으로 휴일이 종료 날짜 이전의 근무일 중에 발생하므로 프로젝트 3의 종료 날짜가 1일 뒤로 시프트됩니다.

다음 캘린더는 프로젝트 3의 시작 날짜와 종료 날짜를 보여 주고 휴일이 프로젝트의 종료 날짜가 1일 변경됨을 보여 줍니다.

5월 18일을 휴일로 하여 프로젝트 3의 시작 날짜와 종료 날짜를 표시하는 캘린더

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

이 함수는 지정된 표준 시간대에 대한 현재 시간의 타임스탬프를 반환합니다.

구문:

```
LocalTime([timezone [, ignoreDST ]])
```

반환 데이터 유형: dual

인수

인수	설명
timezone	**timezone**은 **Date and Time**에 대해 **Windows Control Panel**의 **Time Zone** 아래에 나열된 지리적 위치가 포함된 문자열 또는 'GMT+hh:mm' 형식의 문자열로 지정됩니다. 허용되는 장소 및 표준 시간대 목록도 아래 표에 나열되어 있습니다. 표준 시간대를 지정하지 않으면 로컬 시간이 반환됩니다. *DST 오프셋을 사용하는 경우(즉, False로 평가되는 **ignoreDST** 인수 값 지정) **place** 인수에 GMT 오프셋이 아닌 위치를 지정해야 합니다. 이는 일광 절약 시간을 조정하려면 GMT 오프셋에서 제공하는 경도 정보와 함께 위도 정보가 필요하기 때문입니다. 자세한 내용은 DST와 함께 GMT 오프셋 사용 (page 814)을 참조하십시오.*
ignoreDST	이 인수가 True로 평가되면 DST(일광 절약 시간)가 무시됩니다. True로 평가되는 유효한 인수 값에는 -1 및 `True()`이 포함됩니다. 이 인수가 False로 평가되면 일광 절약 시간에 맞게 타임스탬프가 조정됩니다. False로 평가되는 유효한 인수 값에는 0 및 `False()`이 포함됩니다. **ignoreDST** 인수 값이 잘못된 경우 함수는 **ignore_dst** 값이 True로 평가되는 것처럼 표현식을 평가합니다. **ignoreDST** 인수 값이 지정되지 않은 경우 함수는 **ignore_dst** 값이 False로 평가되는 것처럼 표현식을 평가합니다.

유효한 장소 및 표준 시간대

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
Atlantic Time (Canada)	Greenland	Mazatlan	Skopje
Auckland	Greenwich Mean Time : Dublin	Melbourne	Sofia

A-C	D-K	L-R	S-Z
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	-
Chihuahua	-	Riyadh	-
Chongqing	-	Rome	-
Copenhagen	-	-	-

예 및 결과:

아래 예는 로컬 시간 2023-08-14 08:39:47에 호출되는 함수를 기반으로 하며, 서버 또는 데스크톱 환경의 로컬 표준 시간대는 GMT-05:00이고 이 나열된 날짜를 기준으로 일광 절약 시간을 구현한 지역에 있습니다.

스크립팅 예

예	결과
`localtime ()`	로컬 시간 2023-08-14 08:39:47을 반환합니다.
`localtime ('London')`	런던의 로컬 시간인 2023-08-14 13:39:47을 반환합니다.
`localtime ('GMT+02:00')`	GMT+02:00 표준 시간대의 로컬 시간 2023-08-14 14:39:47을 반환합니다. 장소가 아닌 GMT 오프셋이 지정되기 때문에 일광 절약 시간에 대한 조정이 이루어지지 않습니다.
`localtime ('Paris',-1)`	일광 절약 시간을 무시하고 파리의 로컬 시간 2023-08-14 13:39:47을 반환합니다.
`localtime ('Paris',True())`	일광 절약 시간을 무시하고 파리의 로컬 시간 2023-08-14 13:39:47을 반환합니다.
`localtime ('Paris',0)`	일광 절약 시간을 고려하여 파리의 로컬 시간 2023-08-14 14:39:47을 반환합니다.
`localtime ('Paris',False ())`	일광 절약 시간을 고려하여 파리의 로컬 시간 2023-08-14 14:39:47을 반환합니다.

DST와 함께 GMT 오프셋 사용

Qlik Sense에서 ICU(International Components for Unicode) 라이브러리를 구현한 후 DST(일광 절약 시간)와 함께 GMT(그리니치 표준시) 오프셋을 사용하려면 추가 위도 정보가 필요합니다.

GMT는 경도(동-서) 오프셋인이며, DST는 위도(북-남) 오프셋입니다. 예를 들어, 헬싱키(핀란드)와 요하네스버그(남아프리카)는 동일한 GMT+02:00 오프셋을 공유하지만 동일한 DST 오프셋을 공유하지 않습니다. 즉, GMT 오프셋 외에도 모든 DST 오프셋에는 로컬 DST 조건에 대한 전체 정보를 얻기 위해 로컬 표준 시간대의 위도 위치(지리적 표준 시간대 입력)에 대한 정보가 필요합니다.

lunarweekend

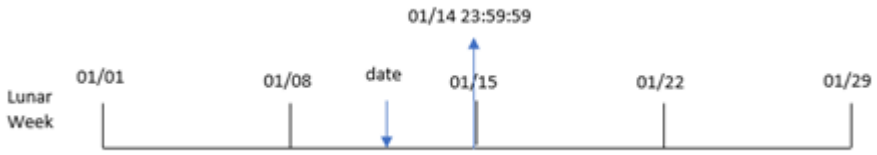
이 함수는 **date**를 포함하는 음력 주의 마지막 날의 마지막 밀리초의 타임스탬프에 해당하는 값을 반환합니다. Qlik Sense에서 음력 주는 1월 1일을 주의 첫 번째 날로 계산하여 정의되며, 연도의 마지막 주를 제외하고 정확히 7일이 포함됩니다.

구문:

```
LunarweekEnd(date[, period_no[, first_week_day]])
```

반환 데이터 유형: dual

lunarweekend() 함수의 다이어그램 예



lunarweekend() 함수는 date가 속하는 음력 주를 확인합니다. 그런 다음 해당 주의 마지막 밀리초에 대한 타임스탬프를 날짜 서식으로 반환합니다.

인수

인수	설명
date	평가할 날짜 또는 타임스탬프입니다.
period_no	**period_no**는 정수 또는 정수로 처리되는 표현식이며, 값 0은 **date**를 포함하는 음력 주를 나타냅니다. **period_no**가 음수 값일 경우 이전 음력 주, 양수 값일 경우 다음 음력 주를 나타냅니다.
first_week_day	0보다 크거나 작을 수 있는 오프셋입니다. 이 함수는 지정된 일수 및/또는 일의 분위수에 따른 연도 시작 날짜를 변경합니다.

사용 시기

lunarweekend() 함수는 일반적으로 사용자가 아직 발생하지 않은 주의 부분을 사용하여 계산하려고 할 때 표현식의 일부로 사용됩니다. weekend() 함수와 달리 각 캘린더 연도의 마지막 음력 주는 12월 31일에 끝납니다. 예를 들어, lunarweekend() 함수를 사용하여 해당 주에 아직 발생하지 않은 이자를 계산할 수 있습니다.

함수 예

예	결과
lunarweekend('01/12/2013')	01/14/2013 23:59:59를 반환합니다.
lunarweekend('01/12/2013', -1)	01/07/2013 23:59:59를 반환합니다.
lunarweekend('01/12/2013', 0, 1)	01/15/2013 23:59:59를 반환합니다.

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

예 1 – 추가 인수 없음

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- `Transactions`라는 테이블에 로드되는 2022년 트랜잭션 집합이 포함된 데이터 집합.
- `DateFormat` 시스템 변수(MM/DD/YYYY) 서식으로 제공된 날짜 필드.
- 트랜잭션이 발생한 음력 주의 끝에 대한 타임스탬프를 반환하는 필드 `end_of_week` 만들기.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- end_of_week
- end_of_week_timestamp

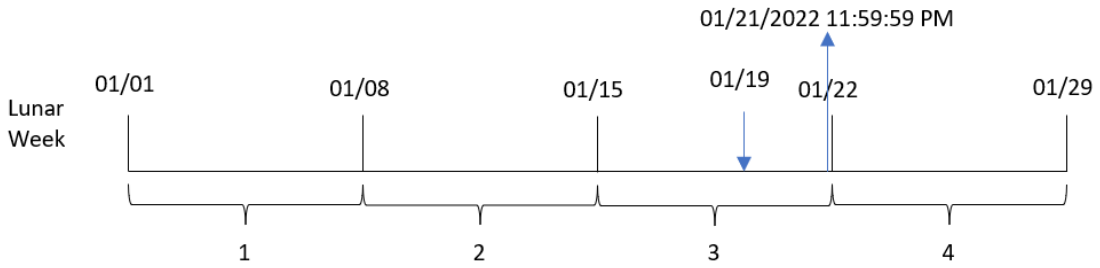
결과 테이블

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
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

end_of_week 필드는 lunarweekend() 함수를 사용하고 date 필드를 함수의 인수로 전달하여 선행 LOAD 문에서 만들어집니다.

lunarweekend() 함수는 날짜 값이 속하는 음력 주를 식별하고 해당 주의 마지막 밀리초에 대한 타임스탬프를 반환합니다.

lunarweekend() 함수 다이어그램, 추가 인수가 없는 예



트랜잭션 8189는 1월 19일에 발생했습니다. lunarweekend() 함수는 음력 주가 1월 15일에 시작함을 식별합니다. 따라서 해당 트랜잭션의 end_of_week 값은 음력 주의 마지막 밀리초인 1월 21일 오후 11:59:59를 반환합니다.

예 2 – period_no

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 첫 번째 예와 동일한 데이터 집합 및 시나리오.
- 트랜잭션이 발생하기 전 음력 주의 끝에 대한 타임스탬프를 반환하는 필드 previous_lunar_week_end 만들기.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- previous_lunar_week_end
- previous_lunar_week_end_timestamp

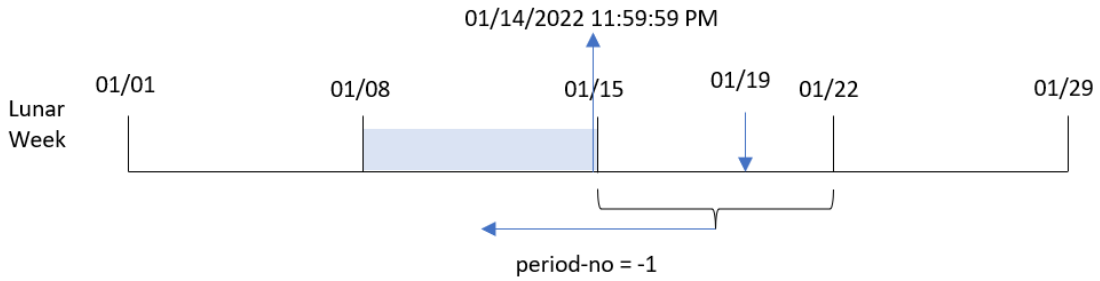
결과 테이블

date	previous_lunar_week_end	previous_lunar_week_end_timestamp
1/7/2022	12/31/2021	12/31/2021 11:59:59 PM
1/19/2022	01/14/2022	1/14/2022 11:59:59 PM
2/5/2022	02/04/2022	2/4/2022 11:59:59 PM
2/28/2022	02/25/2022	2/25/2022 11:59:59 PM
3/16/2022	03/11/2022	3/18/2022 11:59:59 PM
4/1/2022	03/25/2022	3/25/2022 11:59:59 PM
5/7/2022	05/06/2022	5/6/2022 11:59:59 PM
5/16/2022	05/13/2022	5/13/2022 11:59:59 PM
6/15/2022	06/10/2022	6/10/2022 11:59:59 PM
6/26/2022	06/24/2022	6/24/2022 11:59:59 PM
7/9/2022	07/08/2022	7/8/2022 11:59:59 PM
7/22/2022	07/15/2022	7/15/2022 11:59:59 PM
7/23/2022	07/22/2022	7/22/2022 11:59:59 PM
7/27/2022	07/22/2022	7/22/2022 11:59:59 PM

date	previous_lunar_week_end	previous_lunar_week_end_timestamp
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

이 경우 lunarweekend() 함수에서 period_no -1을 오프셋 인수로 사용했으므로 이 함수는 먼저 트랜잭션이 발생한 음력 주를 식별합니다. 그런 다음 1주일 전으로 시프트하여 해당 음력 주의 마지막 밀리초를 식별합니다.

lunarweekend() 함수의 다이어그램, period_no 예



트랜잭션 8189는 1월 19일에 발생했습니다. lunarweekend() 함수는 음력 주가 1월 15일에 시작함을 식별합니다. 따라서 이전 음력 주는 1월 8일에 시작하여 1월 14일 오후 11:59:59에 끝났습니다. previous_lunar_week_end 필드에 대해 반환되는 값입니다.

예 3 – first_week_day

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

이 로드 스크립트에는 첫 번째 예와 동일한 데이터 집합 및 시나리오가 포함되어 있습니다. 이 예에서는 음력 주를 1월 5일에 시작하도록 설정했습니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- end_of_week
- end_of_week_timestamp

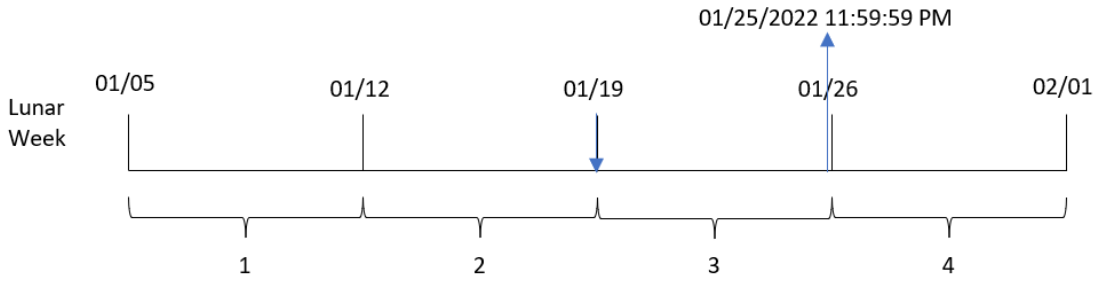
결과 테이블

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

date	end_of_week	end_of_week_timestamp
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

이 경우 `lunarweekend()` 함수에서 `first_week_date` 인수가 4로 사용되므로 연도의 시작을 1월 1일에서 1월 5일로 오프셋합니다.

`lunarweekend()` 함수 다이어그램, first_week_day 예



트랜잭션 8189는 1월 19일에 발생했습니다. 음력 주가 1월 5일에 시작하므로 `lunarweekend()` 함수는 1월 19일을 포함하는 음력 주가 1월 19일에도 시작함을 식별합니다. 따라서 해당 음력 주는 1월 25일 오후 11:59:59에 끝납니다. 이는 `end_of_week` 필드에 대해 반환된 값입니다.

예 4 – 차트 개체 예

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

이 로드 스크립트에는 첫 번째 예와 동일한 데이터 집합 및 시나리오가 포함되어 있습니다.

그러나 이 예에서는 변경되지 않은 데이터 집합이 응용 프로그램에 로드됩니다. 트랜잭션이 발생한 음력 주의 끝에 대한 타임스탬프를 반환하는 계산은 응용 프로그램의 차트 개체에서 측정값으로 만들어집니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다. date.

다음 측정값을 추가합니다.

```
=lunarweekend(date)
```

```
=timestamp(lunarweekend(date))
```

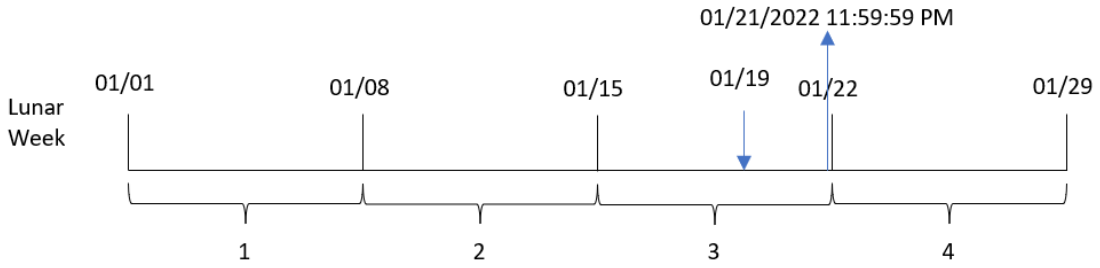
결과 테이블

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
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

end_of_week 측정값은 lunarweekend() 함수를 사용하고 date 필드를 함수의 인수로 전달하여 차트 개체에서 만들어집니다.

lunarweekend() 함수는 날짜 값이 속하는 음력 주를 식별하고 해당 주의 마지막 밀리초에 대한 타임스탬프를 반환합니다.

lunarweekend() 함수의 다이어그램, 차트 개체 예



트랜잭션 8189는 1월 19일에 발생했습니다. lunarweekend() 함수는 음력 주가 1월 15일에 시작함을 식별합니다. 따라서 해당 트랜잭션의 end_of_week 값은 음력 주의 마지막 밀리초인 1월 21일 오후 11:59:59를 반환합니다.

예 5 – 시나리오

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Employee_Expenses라는 테이블에 로드되는 데이터 집합.
- 직원 ID, 직원 이름 및 각 직원의 평균 일일 비용 청구입니다.

최종 사용자는 직원 ID와 직원 이름별로 해당 음력 주에 남은 기간 동안 발생할 것으로 예상되는 비용 청구를 표시하는 차트 개체를 원합니다.

로드 스크립트

```
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
];
```

결과

다음과 같이 하십시오.

1. 데이터를 로드하고 시트를 엽니다. 새 테이블을 만듭니다.
2. 다음 필드를 차원으로 추가합니다.
    - employee_id
    - employee_name
3. 그런 다음 누적된 이자를 계산하기 위해 다음 측정값을 만듭니다.
   =(lunarweekend(today(1))-today(1))*avg_daily_claim
4. 측정값의 **숫자 형식**을 **화폐**로 설정합니다.

결과 테이블

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

lunarkweekend() 함수는 오늘 날짜를 유일한 인수로 사용하여 현재 음력 주의 종료 날짜를 반환합니다. 그런 다음 이 표현식은 음력 주 끝 날짜에서 오늘 날짜를 빼서 이번 주에 남아 있는 일 수를 반환합니다.

그런 다음 이 값에 각 직원의 평균 일일 비용 클레임을 곱하여 각 직원이 남은 음력 주에 예상되는 청구 금액을 계산합니다.

lunarweekname

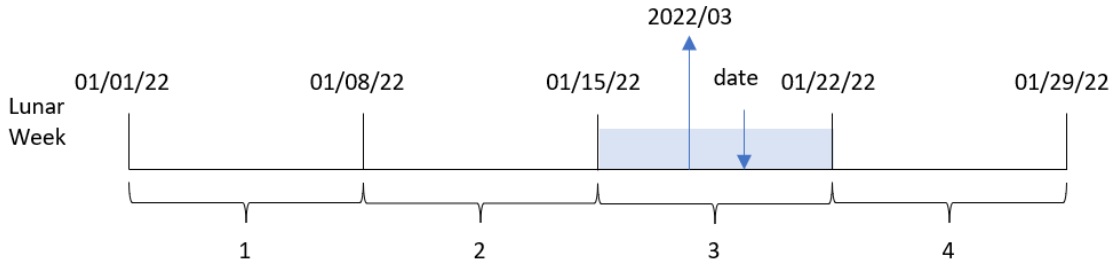
이 함수는 **date**를 포함한 음력 주의 첫 번째 날의 첫 번째 밀리초의 타임스탬프에 해당하는 음력 주차와 연도를 보여주는 표시 값을 반환합니다. Qlik Sense에서 음력 주는 1월 1일을 주의 첫 번째 날로 계산하여 정의되며, 연도의 마지막 주를 제외하고 정확히 7일이 포함됩니다.

구문:

```
LunarWeekName(date [, period_no[, first_week_day]])
```

반환 데이터 유형: dual

lunarweekname() 함수의 다이어그램 예



lunarweekname() 함수는 1월 1일부터 주 계산을 시작하여 날짜가 속하는 음력 주를 확인합니다. 그런 다음 year/weekcount로 구성된 값을 반환합니다.

인수

인수	설명
date	평가할 날짜 또는 타임스탬프입니다.
period_no	**period_no**는 정수 또는 정수로 처리되는 표현식이며, 값 0은 **date**를 포함하는 음력 주를 나타냅니다. **period_no**가 음수 값일 경우 이전 음력 주, 양수 값일 경우 다음 음력 주를 나타냅니다.
first_week_day	0보다 크거나 작을 수 있는 오프셋입니다. 이 함수는 지정된 일수 및/또는 일의 분위수에 따른 연도 시작 날짜를 변경합니다.

사용 시기

lunarweekname() 함수는 음력 주별로 집계를 비교하려는 경우에 유용합니다. 예를 들어, 이 함수를 사용하여 음력 주별로 제품의 총 판매량을 확인할 수 있습니다. 음력 주는 연도의 첫 번째 주에 포함된 모든 값에 빠르면 1월 1일의 값만 포함해야 하는 경우에 유용합니다.

이러한 차원은 마스터 캘린더 테이블에 필드를 만드는 함수를 사용하여 로드 스크립트에서 만들 수 있습니다. 이 함수는 차트에서 계산된 차원으로 직접 사용할 수도 있습니다.

함수 예

예	결과
`lunarweekname('01/12/2013')`	2006/02를 반환합니다.
`lunarweekname('01/12/2013', -1)`	2006/01를 반환합니다.
`lunarweekname('01/12/2013', 0, 1)`	2006/02을 반환합니다.

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

예 1 – 추가 인수가 없는 날짜

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Transactions라는 테이블에 로드되는 2022년 트랜잭션 집합이 포함된 데이터 집합.
- DateFormat 시스템 변수(MM/DD/YYYY) 서식으로 제공된 날짜 필드.
- 트랜잭션이 발생한 음력 주에 대한 연도 및 주차를 반환하는 필드 lunar_week_name 만들기.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- lunar_week_name

결과 테이블

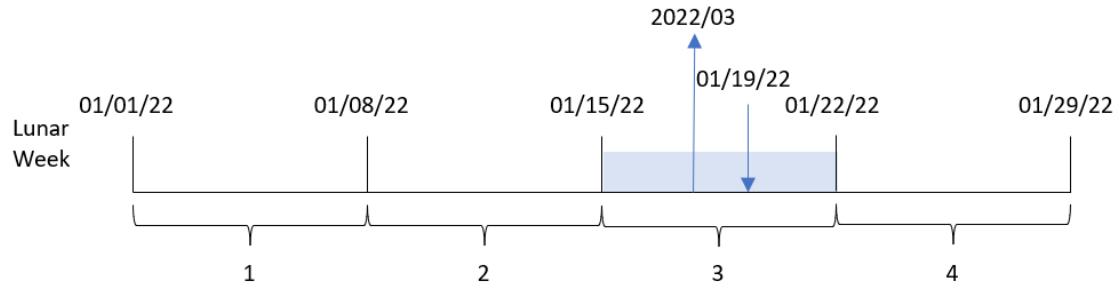
date	lunar_week_name
1/7/2022	2022/01
1/19/2022	2022/03
2/5/2022	2022/06
2/28/2022	2022/09
3/16/2022	2022/11
4/1/2022	2022/13
5/7/2022	2022/19
5/16/2022	2022/20
6/15/2022	2022/24
6/26/2022	2022/26
7/9/2022	2022/28
7/22/2022	2022/29
7/23/2022	2022/30
7/27/2022	2022/30
8/2/2022	2022/31
8/8/2022	2022/32
8/19/2022	2022/33
9/26/2022	2022/39

date	lunar_week_name
10/14/2022	2022/41
10/29/2022	2022/44

lunar_week_name 필드는 lunarweekname() 함수를 사용하고 date 필드를 함수의 인수로 전달하여 선행 LOAD 문에서 만들어집니다.

lunarweekname() 함수는 날짜 값이 속하는 음력 주를 식별하여 해당 날짜의 연도와 주차를 반환합니다.

lunarweekname() 함수 다이어그램, 추가 인수가 없는 예



트랜잭션 8189는 1월 19일에 발생했습니다. lunarweekname() 함수는 이 날짜가 1월 15일에 시작하는 음력 주에 해당한다는 것을 식별합니다. 이는 올해의 세 번째 음력 주입니다. 따라서 해당 트랜잭션에 대해 반환된 lunar_week_name 값은 2022/03입니다.

예 2 – period_no 인수를 사용한 날짜

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 첫 번째 예와 동일한 데이터 집합 및 시나리오.
- 트랜잭션이 발생한 이전 음력 주에 대한 연도 및 주차를 반환하는 필드 previous_lunar_week_name 만들기.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- previous_lunar_week_name

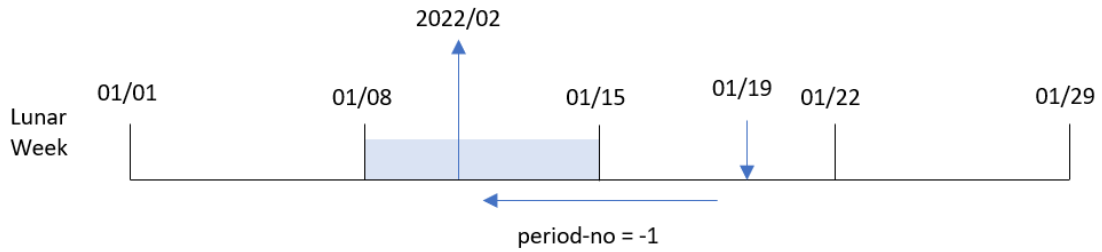
결과 테이블

date	previous_lunar_week_name
1/7/2022	2021/52
1/19/2022	2022/02
2/5/2022	2022/05
2/28/2022	2022/08
3/16/2022	2022/10
4/1/2022	2022/12
5/7/2022	2022/18
5/16/2022	2022/19
6/15/2022	2022/23
6/26/2022	2022/25

date	previous_lunar_week_name
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

이 경우 `lunarweekname()` 함수에서 `period_no -1`을 오프셋 인수로 사용했으므로 이 함수는 먼저 트랜잭션이 발생한 음력 주를 식별합니다. 그런 다음 연도와 일주일 전의 숫자를 반환합니다.

`lunarweekname()` 함수의 다이어그램, `period_no` 예



트랜잭션 8189는 1월 19일에 발생했습니다. `lunarweekname()` 함수는 이 트랜잭션이 해당 연도의 세 번째 음력 주에 발생했음을 식별하므로 `previous_lunar_week_name` 필드에 일주일 전의 연도 및 값 2022/02를 반환합니다.

예 3 – first_week_day 인수를 사용한 날짜

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

이 로드 스크립트에는 첫 번째 예와 동일한 데이터 집합 및 시나리오가 포함되어 있습니다. 이 예에서는 음력 주를 1월 5일에 시작하도록 설정했습니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

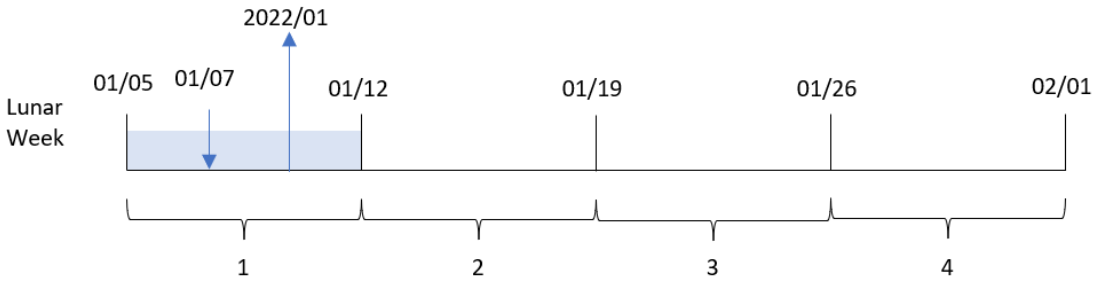
- date
- lunar_week_name

결과 테이블

date	lunar_week_name
1/7/2022	2022/01
1/19/2022	2022/03
2/5/2022	2022/05
2/28/2022	2022/08

date	lunar_week_name
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

lunarweekname() 함수 다이어그램, first_week_day 예



이 경우 lunarweekname() 함수에서 first_week_date 인수가 4로 사용되므로 음력 주의 시작을 1월 1일에서 1월 5일로 오프셋합니다.

트랜잭션 8188은 1월 7일에 발생했습니다. 1월 5일에 시작하는 음력 주로 인해 lunarweekname() 함수는 1월 7일을 포함하는 음력 주가 해당 연도의 첫 번째 음력 주임을 식별합니다. 따라서 해당 트랜잭션에 대해 반환된 lunar_week_name 값은 2022/01입니다.

예 4 – 차트 개체 예

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

이 로드 스크립트에는 첫 번째 예와 동일한 데이터 집합 및 시나리오가 포함되어 있습니다.

그러나 이 예에서는 변경되지 않은 데이터 집합이 응용 프로그램에 로드됩니다. 트랜잭션이 발생한 음력 주차와 연도를 반환하는 계산은 응용 프로그램의 차트 개체에서 측정값으로 만들어집니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다. `date`.

트랜잭션이 발생한 음력 주의 시작 날짜를 계산하려면 다음 측정값을 만듭니다.

```
=lunarweekname(date)
```

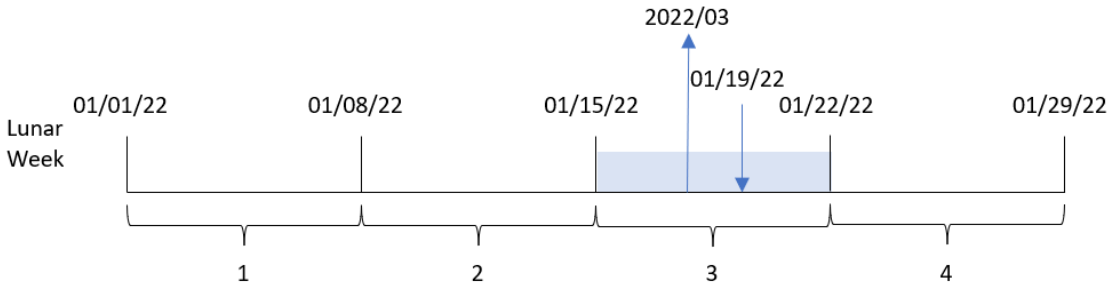
결과 테이블

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

`lunar_week_name` 측정값은 `lunarweekname()` 함수를 사용하고 `date` 필드를 함수의 인수로 전달하여 차트 개체에서 만들어집니다.

`lunarweekname()` 함수는 날짜 값이 속하는 음력 주를 식별하여 해당 날짜의 연도와 주차를 반환합니다.

lunarweekname() 함수의 다이어그램, 차트 개체 예



트랜잭션 8189는 1월 19일에 발생했습니다. lunarweekname() 함수는 이 날짜가 1월 15일에 시작하는 음력 주에 해당한다는 것을 식별합니다. 이는 올해의 세 번째 음력 주입니다. 따라서 해당 트랜잭션의 lunar_week_name 값은 2022/03입니다.

예 5 – 시나리오

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Transactions라는 테이블에 로드되는 2022년 트랜잭션 집합이 포함된 데이터 집합.
- DateFormat 시스템 변수(MM/DD/YYYY) 서식으로 제공된 날짜 필드.

최종 사용자는 현재 연도의 주별 총 판매액을 표시하는 차트 개체를 원합니다. 7일 길이의 1주차는 1월 1일에 시작해야 합니다. 이는 데이터 모델에서 이 차원을 사용할 수 없는 경우에도 차트에서 lunarweekname() 함수를 계산된 차원으로 사용하여 달성할 수 있습니다.

로드 스크립트

```
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
];
```

결과

다음과 같이 하십시오.

1. 데이터를 로드하고 시트를 엽니다. 새 테이블을 만듭니다.
2. 다음 표현식을 사용하여 계산된 차원을 만듭니다.
   `=lunarweekname(date)`
3. 다음 집계 측정값을 사용하여 총 판매액을 계산합니다.
   `=sum(amount)`
4. 측정값의 **숫자 형식**을 **화폐**로 설정합니다.

결과 테이블

=lunarweekname(date)	=sum(amount)
2022/01	$17.17
2022/03	$37.23
2022/06	$57.42
2022/09	$88.27
2022/11	$53.80
2022/13	$82.06
2022/19	$40.39
2022/20	$87.21
2022/24	$95.93
2022/26	$45.89
2022/28	$36.23
2022/29	$25.66
2022/30	$152.75
2022/31	$76.11

=lunarweekname(date)	=sum(amount)
2022/32	$25.12
2022/33	$46.23
2022/39	$84.21
2022/41	$96.24
2022/44	$67.67

lunarweekstart

이 함수는 **date**를 포함하는 음력 주의 첫 번째 날의 첫 번째 밀리초의 타임스탬프에 해당하는 값을 반환합니다. Qlik Sense에서 음력 주는 1월 1일을 주의 첫 번째 날로 계산하여 정의되며, 연도의 마지막 주를 제외하고 정확히 7일이 포함됩니다.

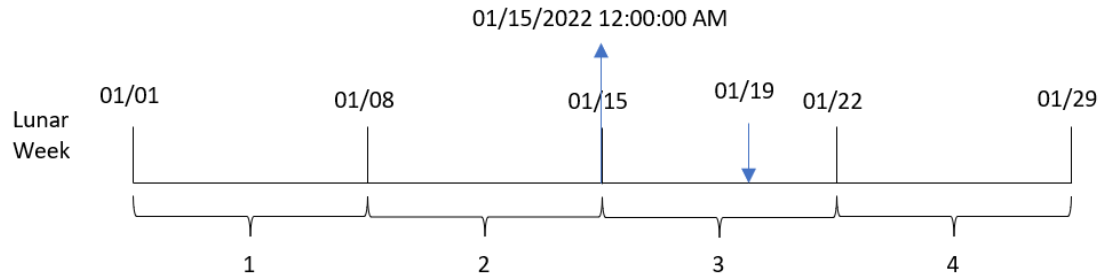
구문:

```
LunarweekStart(date[, period_no[, first_week_day]])
```

반환 데이터 유형: dual

`lunarweekstart()` 함수는 `date`가 속하는 음력 주를 확인합니다. 그런 다음 해당 주의 첫 번째 밀리초에 대한 타임스탬프를 날짜 서식으로 반환합니다.

`lunarweekstart()` 함수의 다이어그램 예



인수

인수	설명
date	평가할 날짜 또는 타임스탬프입니다.
period_no	**period_no**는 정수 또는 정수로 처리되는 표현식이며, 값 0은 **date**를 포함하는 음력 주를 나타냅니다. **period_no**가 음수 값일 경우 이전 음력 주, 양수 값일 경우 다음 음력 주를 나타냅니다.
first_week_day	0보다 크거나 작을 수 있는 오프셋입니다. 이 함수는 지정된 일수 및/또는 일의 분위수에 따른 연도 시작 날짜를 변경합니다.

사용 시기

lunarweekstart() 함수는 일반적으로 사용자가 지금까지 경과된 주의 부분을 사용하여 계산하려고 할 때 표현식의 일부로 사용됩니다. weekstart() 함수와 달리, 새로운 캘린더 연도가 시작될 때마다 주는 1월 1일에 시작하고 다음 주는 7일 후에 시작됩니다. lunarweekstart() 함수는 FirstWeekDay 시스템 변수의 영향을 받지 않습니다.

예를 들어, lunarweekstart()는 해당 주부터 현재까지 누적된 이자를 계산하는 데 사용할 수 있습니다.

함수 예

예	결과
lunarweekstart ('01/12/2013')	01/08/2013를 반환합니다.
lunarweekstart ('01/12/2013', -1)	01/01/2013을 반환합니다.
lunarweekstart ('01/12/2013', 0, 1)	first_week_day를 1로 설정하면 연도의 시작이 01/02/2013으로 변경됨을 의미하므로 01/09/2013을 반환합니다.

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

예 1 – 추가 인수 없음

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Transactions라는 테이블에 로드되는 2022년 트랜잭션 집합이 포함된 데이터 집합.
- DateFormat 시스템 변수(MM/DD/YYYY) 서식으로 제공된 날짜 필드.
- 트랜잭션이 발생한 음력 주의 시작에 대한 타임스탬프를 반환하는 필드 start_of_week 만들기.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- start_of_week
- start_of_week_timestamp

결과 테이블

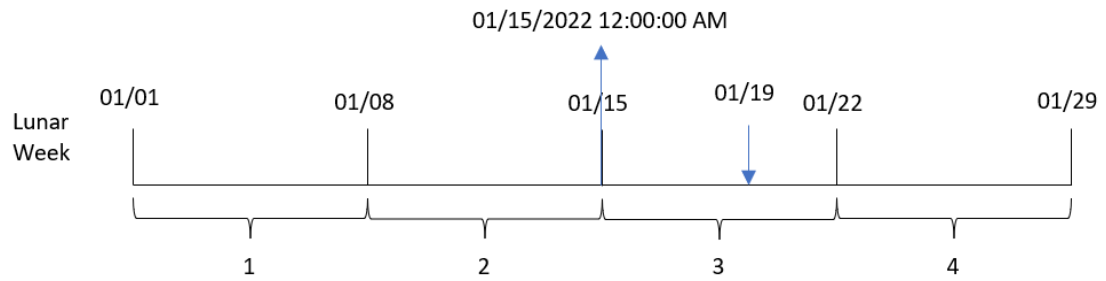
date	start_of_week	start_of_week_timestamp
1/7/2022	01/01/2022	1/1/2022 12:00:00 AM
1/19/2022	01/15/2022	1/15/2022 12:00:00 AM

date	start_of_week	start_of_week_timestamp
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

start_of_week 필드는 lunarweekstart() 함수를 사용하고 date 필드를 함수의 인수로 전달하여 선행 LOAD 문에서 만들어집니다.

lunarweekstart() 함수는 날짜가 속하는 음력 주를 식별하여 해당 주의 첫 번째 밀리초에 대한 타임스탬프를 반환합니다.

lunarweekstart() 함수 다이어그램, 추가 인수가 없는 예



트랜잭션 8189는 1월 19일에 발생했습니다. lunarweekstart() 함수는 음력 주가 1월 15일에 시작함을 식별합니다. 따라서 해당 트랜잭션의 start_of_week 값은 해당 날짜의 첫 번째 밀리초, 즉 1월 15일 오전 12:00:00를 반환합니다.

예 2 – period_no

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 첫 번째 예와 동일한 데이터 집합 및 시나리오.
- 트랜잭션이 발생하기 전 음력 주의 시작에 대한 타임스탬프를 반환하는 필드 previous_lunar_week_start 만들기.

로드 스크립트

```
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
];
```

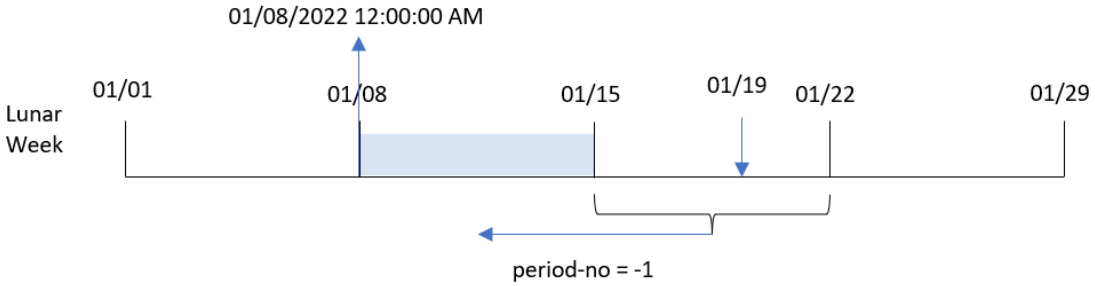
결과

결과 테이블

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

이 경우 `lunarweekstart()` 함수에서 `period_no -1`을 오프셋 인수로 사용했으므로 이 함수는 먼저 트랜잭션이 발생한 음력 주를 식별합니다. 그런 다음 일주일 전으로 시프트하여 해당 음력 주의 첫 번째 밀리초를 식별합니다.

lunarweekstart() 함수의 다이어그램, period_no 예



트랜잭션 8189는 1월 19일에 발생했습니다. lunarweekstart() 함수는 음력 주가 1월 15일에 시작함을 식별합니다. 따라서 이전 음력 주는 1월 8일 오전 12:00:00에 시작되었습니다. 이는 previous_lunar_week_start 필드에 대해 반환된 값입니다.

예 3 – first_week_day

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

이 로드 스크립트에는 첫 번째 예와 동일한 데이터 집합 및 시나리오가 포함되어 있습니다. 이 예에서는 음력 주를 1월 5일에 시작하도록 설정했습니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- start_of_week
- start_of_week_timestamp

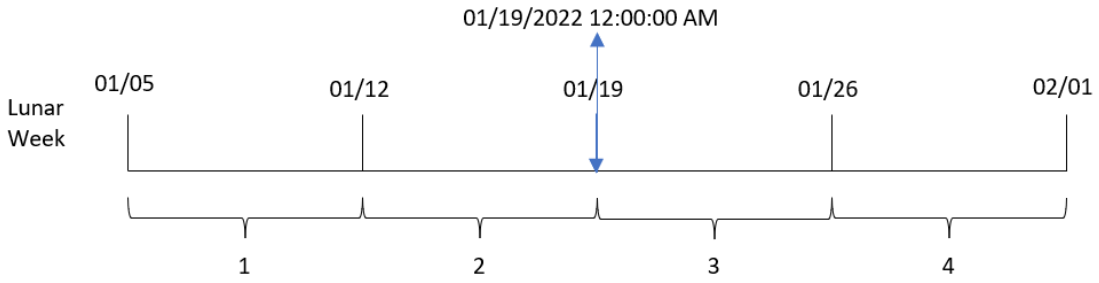
결과 테이블

date	start_of_week	start_of_week_timestamp
1/7/2022	01/05/2022	1/5/2022 12:00:00 AM
1/19/2022	01/19/2022	1/19/2022 12:00:00 AM
2/5/2022	02/02/2022	2/2/2022 12:00:00 AM
2/28/2022	02/23/2022	2/23/2022 12:00:00 AM
3/16/2022	03/16/2022	3/16/2022 12:00:00 AM
4/1/2022	03/30/2022	3/30/2022 12:00:00 AM
5/7/2022	05/04/2022	5/4/2022 12:00:00 AM
5/16/2022	05/11/2022	5/11/2022 12:00:00 AM
6/15/2022	06/15/2022	6/15/2022 12:00:00 AM
6/26/2022	06/22/2022	6/22/2022 12:00:00 AM
7/9/2022	07/06/2022	7/6/2022 12:00:00 AM
7/22/2022	07/20/2022	7/20/2022 12:00:00 AM
7/23/2022	07/20/2022	7/20/2022 12:00:00 AM
7/27/2022	07/27/2022	7/27/2022 12:00:00 AM
8/2/2022	07/27/2022	7/27/2022 12:00:00 AM
8/8/2022	08/03/2022	8/3/2022 12:00:00 AM
8/19/2022	08/17/2022	8/17/2022 12:00:00 AM

date	start_of_week	start_of_week_timestamp
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

이 경우 lunarweekstart() 함수에서 first_week_date 인수가 4로 사용되므로 연도의 시작을 1월 1일에서 1월 5일로 오프셋합니다.

lunarweekstart() 함수 다이어그램, first_week_day 예



트랜잭션 8189는 1월 19일에 발생했습니다. 음력 주가 1월 5일에 시작하므로 lunarweekstart() 함수는 1월 19일을 포함하는 음력 주가 1월 19일 오전 12:00:00에도 시작함을 식별합니다. 따라서 start_of_week 필드에 대해 반환된 값입니다.

예 4 – 차트 개체 예

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

이 로드 스크립트에는 첫 번째 예와 동일한 데이터 집합 및 시나리오가 포함되어 있습니다.

그러나 이 예에서는 변경되지 않은 데이터 집합이 응용 프로그램에 로드됩니다. 트랜잭션이 발생한 음력 주의 시작에 대한 타임스탬프를 반환하는 계산은 응용 프로그램의 차트 개체에서 측정값으로 만들어집니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다. date.

다음 측정값을 추가합니다.

```
=lunarweekstart(date)
```

```
=timestamp(lunarweekstart(date))
```

결과 테이블

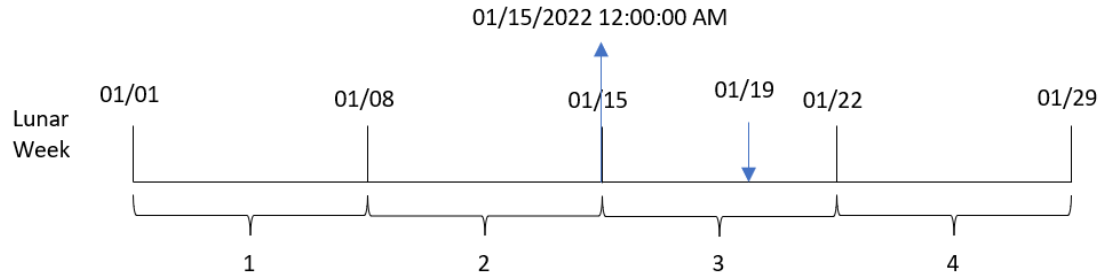
date	=lunarweekstart(date)	=timestamp(lunarweekstart(date))
1/7/2022	01/01/2022	1/1/2022 12:00:00 AM
1/19/2022	01/15/2022	1/15/2022 12:00:00 AM
2/5/2022	02/05/2022	2/5/2022 12:00:00 AM
2/28/2022	02/26/2022	2/26/2022 12:00:00 AM
3/16/2022	03/12/2022	3/12/2022 12:00:00 AM
4/1/2022	03/26/2022	3/26/2022 12:00:00 AM
5/7/2022	05/07/2022	5/7/2022 12:00:00 AM
5/16/2022	05/14/2022	5/14/2022 12:00:00 AM
6/15/2022	06/11/2022	6/11/2022 12:00:00 AM
6/26/2022	06/25/2022	6/25/2022 12:00:00 AM
7/9/2022	07/09/2022	7/9/2022 12:00:00 AM
7/22/2022	07/16/2022	7/16/2022 12:00:00 AM
7/23/2022	07/23/2022	7/23/2022 12:00:00 AM

date	=lunarweekstart(date)	=timestamp(lunarweekstart(date))
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

start_of_week 측정값은 lunarweekstart() 함수를 사용하고 날짜 필드를 함수의 인수로 전달하여 차트 개체에서 만들어집니다.

lunarweekstart() 함수는 날짜 값이 속하는 음력 주를 식별하고 해당 주의 마지막 밀리초에 대한 타임스탬프를 반환합니다.

lunarweekstart() 함수의 다이어그램, 차트 개체 예



트랜잭션 8189는 1월 19일에 발생했습니다. lunarweekstart() 함수는 음력 주가 1월 15일에 시작함을 식별합니다. 따라서 해당 트랜잭션의 start_of_week 값은 해당 날짜의 첫 번째 밀리초, 즉 1월 15일 오전 12:00:00입니다.

예 5 – 시나리오

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Loans라는 테이블에 로드되는 대출 잔액 집합이 포함된 데이터 집합.
- 대출 ID, 주초 잔액, 각 대출에 연간 부과되는 단순 이자율로 구성된 데이터.

최종 사용자는 해당 주부터 현재까지 각 대출에 대해 발생한 현재 이자를 대출 ID별로 표시하는 차트 개체를 원합니다.

로드 스크립트

```
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
];
```

결과

다음과 같이 하십시오.

1. 데이터를 로드하고 시트를 엽니다. 새 테이블을 만듭니다.
2. 다음 필드를 차원으로 추가합니다.
   - loan_id
   - start_balance
3. 그런 다음 누적된 이자를 계산하기 위해 다음 측정값을 만듭니다.
   `=start_balance*(rate*(today(1)-lunarweekstart(today(1)))/365)`
4. 측정값의 **숫자 형식**을 **화폐**로 설정합니다.

결과 테이블

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

오늘 날짜를 유일한 인수로 사용하는 lunarweekstart() 함수는 현재 연도의 시작 날짜를 반환합니다. 이 표현식은 현재 날짜에서 해당 결과를 빼서 이번 주 지금까지 경과한 일 수를 반환합니다.

그런 다음 이 값에 이자율을 곱하고 365로 나누어 이 기간 동안 발생한 유효 이자율을 반환합니다. 그런 다음 결과에 대출의 시작 잔액을 곱하여 이번 주 지금까지 발생한 이자를 반환합니다.

makedate

이 함수는 연도 **YYYY**, 월 **MM**, 일 **DD**에서 계산한 날짜를 반환합니다.

구문:

```
MakeDate(YYYY [ , MM [ , DD ] ])
```

반환 데이터 유형: dual

인수

인수	설명
YYYY	연도는 정수입니다.
MM	월은 정수입니다. 월을 지정하지 않으면 1(1월)이 사용됩니다.
DD	일은 정수입니다. 일을 지정하지 않으면 1(1일)이 사용됩니다.

사용 시기

makedate() 함수는 일반적으로 데이터 생성을 위한 스크립트에서 캘린더를 생성하는 데 사용됩니다. 이는 날짜 필드를 날짜로 직접 사용할 수 없지만 연도, 월 및 일 구성 요소를 추출하기 위해 일부 변환이 필요한 경우에도 사용할 수 있습니다.

이 예에서는 날짜 서식 MM/DD/YYYY를 사용합니다. 날짜 서식은 데이터 로드 스크립트 상단의 SET DateFormat 문에 지정되어 있습니다. 요구 사항에 맞게 예의 형식을 변경하십시오.

함수 예

예	결과
makedate(2012)	01/01/2012를 반환합니다.
makedate(12)	01/01/2012를 반환합니다.
makedate(2012,12)	12/01/2012를 반환합니다.
makedate(2012,2,14)	02/14/2012를 반환합니다.

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

예 1 – 기본 예

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Transactions라는 테이블에 로드되는 2018년 트랜잭션 집합이 포함된 데이터 집합.
- DateFormat 시스템 변수(MM/DD/YYYY) 서식으로 제공된 날짜 필드.
- MM/DD/YYYY 서식의 날짜를 반환하는 필드 transaction_date 만들기.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- transaction_year
- transaction_month
- transaction_day
- transaction_date

결과 테이블

transaction_year	transaction_month	transaction_day	transaction_date
2018	08	30	08/30/2018

transaction_year	transaction_month	transaction_day	transaction_date
2018	09	07	09/07/2018
2018	09	16	09/16/2018
2018	09	22	09/22/2018
2018	09	23	09/23/2018

transaction_date 필드는 makedate() 함수를 사용하고 연도, 월, 일 필드를 함수 인수로 전달하여 선행 LOAD 문에서 만들어집니다.

그런 다음 함수는 이러한 값을 결합하고 날짜 필드로 변환하여 DateFormat 시스템 변수의 형식으로 결과를 반환합니다.

예 2 – 수정된 DateFormat

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 첫 번째 예와 동일한 데이터 집합 및 시나리오.
- DateFormat 시스템 변수를 수정하지 않고 DD/MM/YYYY 서식으로 필드 transaction_date 만들기.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- transaction_year
- transaction_month
- transaction_day
- transaction_date

결과 테이블

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

이 경우 makedate() 함수는 date() 함수 내부에 중첩됩니다. date() 함수의 두 번째 인수는 makedate() 함수 결과의 서식을 필수 DD/MM/YYYY로 설정합니다.

예 3 – 차트 개체 예

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Transactions라는 테이블에 로드되는 2018년 트랜잭션 집합이 포함된 데이터 집합.
- 두 필드에 제공되는 트랜잭션 날짜: year 및 month.

MM/DD/YYYY 서식으로 날짜를 반환하는 차트 개체 측정값 transaction_date를 만듭니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- year
- month

transaction_date를 확인하려면 다음 측정값을 만듭니다.

```
=makedate(transaction_year,transaction_month)
```

결과 테이블

transaction_year	transaction_month	transaction_date
2018	08	08/01/2018
2018	09	09/01/2018

transaction_date 측정값은 makedate() 함수를 사용하고 연도 및 월 필드를 함수 인수로 전달하여 차트 개체에서 만들어집니다.

그런 다음 이 함수는 이러한 값과 가정된 날짜 값 01을 결합합니다. 이러한 값은 날짜 필드로 변환되어 DateFormat 시스템 변수 형식으로 결과를 반환합니다.

예 4 – 시나리오

로드 스크립트 및 차트 표현식

개요

캘린더 연도 2022에 대한 캘린더 데이터 집합을 만듭니다.

로드 스크립트

```
SET DateFormat='MM/DD/YYYY';

Calendar:
       load
               *
    where year(date)=2022;
load
       date(recno()+makedate(2021,12,31)) as date
AutoGenerate 400;
```

결과

결과 테이블

date
01/01/2022
01/02/2022
01/03/2022
01/04/2022
01/05/2022
01/06/2022
01/07/2022
01/08/2022
01/09/2022
01/10/2022
01/11/2022
01/12/2022
01/13/2022
01/14/2022
01/15/2022
01/16/2022
01/17/2022
01/18/2022
01/19/2022
01/20/2022
01/21/2022
01/22/2022
01/23/2022
01/24/2022
01/25/2022
+ 340 추가 행

makedate() 함수는 2021년 12월 31일에 대한 날짜 값을 만듭니다. recno() 함수는 1부터 시작하여 테이블에 로드되는 현재 레코드의 레코드 번호를 제공합니다. 따라서 첫 번째 레코드의 날짜는 2022년 1월 1일입니다. 각각의 연속적인 recno()는 이 날짜를 1씩 증가시킵니다. 이 표현식은 값을 날짜로 변환하는 date() 함

수로 래핑됩니다. 이 과정을 autogenerate 함수에 의해 400번 반복합니다. 마지막으로 선행 LOAD를 사용하여 where 조건을 사용하여 2022년의 날짜만 로드할 수 있습니다. 이 스크립트는 2022년의 모든 날짜가 포함된 캘린더를 생성합니다.

maketime

이 함수는 시간 **hh**, 분 **mm**, 초 **ss**에서 계산한 시간을 반환합니다.

구문:

```
MakeTime(hh [ , mm [ , ss ] ])
```

반환 데이터 유형: dual

인수

인수	설명
hh	시간은 정수입니다.
mm	분은 정수입니다. 분을 지정하지 않으면 00으로 가정됩니다.
ss	초는 정수입니다. 초를 지정하지 않으면 00으로 가정됩니다.

사용 시기

maketime() 함수는 일반적으로 데이터 생성을 위한 스크립트에서 시간 필드를 생성하는 데 사용됩니다. 때때로 시간 필드가 입력 텍스트에서 파생되는 경우 이 함수를 사용해 해당 구성 요소를 사용하여 시간을 구성할 수 있습니다.

이 예에서는 h:mm:ss 시간 서식을 사용합니다. 시간 형식은 데이터 로드 스크립트 상단의 SET TimeFormat문에 지정되어 있습니다. 요구 사항에 맞게 예의 형식을 변경하십시오.

함수 예

예	결과
maketime(22)	22:00:00를 반환합니다.
maketime(22, 17)	22:17:00를 반환합니다.
maketime(22,17,52)	22:17:52을 반환합니다.

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

예 1 – maketime()

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Transactions라는 테이블에 로드되는 트랜잭션 집합이 포함된 데이터 집합.
- 3개 필드에 걸쳐 제공되는 트랜잭션 시간: hours, minutes 및 seconds.
- TimeFormat 시스템 변수 서식으로 시간을 반환하는 transaction_time 필드 만들기.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- transaction_hour
- transaction_minute
- transaction_second
- transaction_time

결과 테이블

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

transaction_time 필드는 maketime() 함수를 사용하고 시, 분 및 초 필드를 함수 인수로 전달하여 선행 LOAD 문에서 만들어집니다.

그런 다음 함수는 이러한 값을 결합하고 시간 필드로 변환하여 TimeFormat 시스템 변수의 시간 서식으로 결과를 반환합니다.

예 2 – time() 함수

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 첫 번째 예와 동일한 데이터 집합 및 시나리오.
- TimeFormat 시스템 변수를 수정하지 않고 결과를 24시간 서식으로 표시하도록 하는 필드 transaction_time 만들기.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- transaction_hour
- transaction_minute
- transaction_second
- transaction_time

결과 테이블

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

이 경우 maketime() 함수는 time() 함수 내부에 중첩됩니다. time() 함수의 두 번째 인수는 maketime() 함수 결과의 서식을 필수 h:mm:ss로 설정합니다.

예 3 – 차트 개체 예

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Transactions라는 테이블에 로드되는 트랜잭션 집합이 포함된 데이터 집합.
- 두 필드에 걸쳐 제공되는 트랜잭션 시간: hours 및 minutes.
- TimeFormat 시스템 변수 서식으로 시간을 반환하는 필드 transaction_time 만들기.

h:mm:ss TT 서식으로 시간을 반환하는 차트 개체 측정값 transaction_time을 만듭니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- transaction_hour
- transaction_minute

transaction_time을 계산하려면 다음 측정값을 만듭니다.

```
=maketime(transaction_hour,transaction_minute)
```

결과 테이블

transaction_hour	transaction_minute	=maketime(transaction_hour, transaction_minute)
2	52	2:52:00 AM
6	32	6:32:00 AM
9	25	9:25:00 AM
12	09	12:09:00 PM
17	55	5:55:00 PM
18	43	6:43:00 PM
21	43	9:43:00 PM

transaction_time 측정값은 maketime() 함수를 사용하고 시간 및 분 필드를 함수 인수로 전달하여 차트 개체에서 만들어집니다.

그런 다음 이 함수는 이러한 값을 결합하고 초는 00으로 가정합니다. 이러한 값은 시간 필드로 변환되어 TimeFormat 시스템 변수 형식으로 결과를 반환합니다.

예 4 – 시나리오

로드 스크립트 및 차트 표현식

개요

2022년 1월의 캘린더 데이터 집합을 8시간 단위로 나누어 만듭니다.

로드 스크립트

```
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;
```

결과

결과 테이블

타임스탬프
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

타임스탬프
1/3/2022 4:00:00 PM
1/4/2022 12:00:00 AM
1/4/2022 8:00:00 AM
1/4/2022 4:00:00 PM
1/5/2022 12:00:00 AM
1/5/2022 8:00:00 AM
1/5/2022 4:00:00 PM
1/6/2022 12:00:00 AM
1/6/2022 8:00:00 AM
1/6/2022 4:00:00 PM
1/7/2022 12:00:00 AM
1/7/2022 8:00:00 AM
1/7/2022 4:00:00 PM
1/8/2022 12:00:00 AM
1/8/2022 8:00:00 AM
1/8/2022 4:00:00 PM
1/9/2022 12:00:00 AM
+ 68 추가 행

초기 autogenerate 함수는 tmpCalendar라는 테이블에 1월의 모든 날짜를 포함하는 캘린더를 만듭니다.

세 개의 레코드가 포함된 두 번째 테이블이 만들어집니다. 각 레코드에 대해 recno() - 1을 수행한 결과(값 0, 1, 2)에 8을 곱합니다. 결과적으로 값 0, 8 16이 생성됩니다. 이러한 값은 분 및 초 값이 0인 maketime() 함수에서 시간 매개 변수로 사용됩니다. 결과적으로 테이블에는 오전 12:00:00, 오전 8:00:00 및 오후 4:00:00의 세 가지 시간 필드가 포함됩니다.

이 테이블은 tmpCalendar 테이블에 조인됩니다. 이 조인의 경우 두 테이블 간에 일치하는 필드가 없으므로 시간 행이 각 날짜 행에 추가됩니다. 결과적으로 각 날짜 행은 이제 각 시간 값으로 세 번 반복됩니다.

마지막으로 tmpCalendar 테이블의 Resident LOAD에서 Calendar 테이블이 만들어집니다. 날짜 및 시간 필드는 연결되고 timestamp() 함수로 래핑되어 타임스탬프 필드를 만듭니다.

그런 다음 tmpCalendar 테이블이 삭제됩니다.

makeweekdate

이 함수는 연도, 주차 및 요일에서 계산된 날짜를 반환합니다.

구문:

```
MakeWeekDate(weekyear [, week [, weekday [, first_week_day [, broken_weeks [, reference_day]]]]])
```

반환 데이터 유형: dual

makeweekdate() 함수는 스크립트 및 차트 함수로 모두 사용할 수 있습니다. 이 함수는 함수에 전달된 매개 변수를 기반으로 날짜를 계산합니다.

인수

인수	설명
weekyear	특정 날짜에 대한 WeekYear() 함수에 의해 정의된 연도, 즉 해당 주차가 속한 연도입니다. *예를 들어, 1주가 이미 전년도 12월에 시작된 경우와 같이 연도는 경우에 따라 캘린더 연도와 다를 수 있습니다.*
week	특정 날짜에 대한 Week() 함수에 의해 정의된 주차입니다. 주차를 지정하지 않으면 1로 간주됩니다.
weekday	해당 날짜에 대한 WeekDay() 함수에 의해 정의된 요일입니다. 0은 주의 첫 번째 요일이고 6은 주의 마지막 요일입니다. 요일을 지정하지 않으면 0으로 가정합니다. *0은 항상 첫 번째 요일을 의미하고 6은 항상 마지막 요일을 의미하지만 해당 하는 요일은 **first_week_day** 매개 변수에 의해 확인됩니다. 생략하면 변수 **FirstWeekDay**의 값이 사용됩니다.* 불가능한 매개 변수 조합과 함께 분리된 주를 사용하면 선택한 연도에 속하지 않는 결과가 발생할 수 있습니다. MakeWeekDate(2021,1,0,6,1) 이 날짜는 지정된 주의 첫째 날(일요일)이므로 'Dec 27 2020'을 반환합니다. 2021년 1월 1일은 금요일이었습니다.
first_week_day	주의 시작 요일을 지정합니다. 생략하면 변수 **FirstWeekDay**의 값이 사용됩니다. **first_week_day**에 가능한 값은 월요일에 0, 화요일에 1, 수요일에 2, 목요일에 3, 금요일에 4, 토요일에 5, 일요일에 6입니다. 시스템 변수에 대한 자세한 내용은 *FirstWeekDay (page 219)*를 참조하십시오.
broken_weeks	**broken_weeks**를 지정하지 않으면 **BrokenWeeks** 변수의 값이 주를 분리할지 여부를 정의하는 데 사용됩니다.

인수	설명
reference_day	**reference_day**를 지정하지 않으면 **ReferenceDay** 변수의 값이 1주차를 정의하기 위한 기준일로 설정할 1월 날짜를 정의하는 데 사용됩니다.

사용 시기

makeweekdate() 함수는 일반적으로 데이터를 생성하여 날짜 목록을 생성하거나 입력 데이터에 연도, 주 및 요일이 제공될 때 날짜를 구성하기 위해 스크립트에서 사용됩니다.

다음 예에서는 다음을 가정합니다.

```
SET FirstWeekDay=0;
SET BrokenWeeks=0;
SET ReferenceDay=4;
```

함수 예

예	결과
makeweekdate(2014,6,6)	반환 값: 02/09/2014
makeweekdate(2014,6,1)	반환 값: 02/04/2014
makeweekdate(2014,6)	02/03/2014을 반환합니다(평일 0 사용).

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

예 1 – day 포함

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Sales라는 테이블에 2022년의 주간 판매 합계가 포함된 데이터 집합.
- 3개 필드에 제공되는 트랜잭션 날짜: year, week 및 sales.

- 측정값 end_of_week를 만드는 데 사용되는 선행 LOAD, makeweekdate() 함수를 사용하여 해당 주의 금요일 날짜를 MM/DD/YYYY 서식으로 반환합니다.

반환된 날짜가 금요일임을 증명하기 위해 end_of_week 표현식도 요일을 표시하는 weekday() 함수로 래핑됩니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- transaction_year
- transaction_week
- end_of_week
- week_day

결과 테이블

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

transaction_year	transaction_week	end_of_week	week_day
2022	06	02/11/2022	Fri
2022	07	02/18/2022	Fri

'end_of_week' 필드는 makeweekdate() 함수를 사용하여 선행 LOAD 문에서 만들어집니다. transaction_year, transaction_week 필드는 함수를 통해 연도 및 주 인수로 전달됩니다. day 인수에는 값 4가 사용됩니다.

그런 다음 함수는 이러한 값을 결합하고 날짜 필드로 변환하여 DateFormat 시스템 변수의 형식으로 결과를 반환합니다.

makeweekdate() 함수와 해당 인수도 weekday() 함수로 래핑되어 week_day 필드를 반환합니다. 위의 표에서 볼 수 있듯이 week_day 필드는 이러한 날짜가 금요일에 발생함을 보여 줍니다.

예 2 – day 제외

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Sales라는 테이블에 2022년의 주간 판매 합계가 포함된 데이터 집합.
- 3개 필드에 제공되는 트랜잭션 날짜: year, week 및 sales.
- makeweekdate() 함수를 사용하여 측정값 first_day_of_week를 만드는 데 사용되는 선행 LOAD. 이는 MM/DD/YYYY 서식으로 해당 주의 월요일 날짜를 반환합니다.

반환된 날짜가 월요일임을 증명하기 위해 first_day_of_week 표현식도 요일을 표시하는 weekday() 함수로 래핑됩니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- transaction_year
- transaction_week
- first_day_of_week
- week_day

결과 테이블

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

first_day_of_week 필드는 makeweekdate() 함수를 사용하여 선행 LOAD 문에서 만들어집니다. transaction_year 및 transaction_week 매개 변수는 함수 인수로 전달되고 day 매개 변수는 비어 있습니다.

그런 다음 함수는 이러한 값을 결합하고 날짜 필드로 변환하여 DateFormat 시스템 변수의 형식으로 결과를 반환합니다.

makeweekdate() 함수와 해당 인수도 week_day 필드를 반환하는 weekday() 함수로 래핑됩니다. 위의 표에서 볼 수 있듯이 makeweekdate() 함수에서 해당 매개 변수가 비어 있기 때문에 week_day 필드는 모든 경우에 월요일을 반환합니다. 여기에서 기본값은 0(첫 번째 요일)이고 첫 번째 요일은 FirstWeekDay 시스템 변수에 의해 월요일로 설정됩니다.

예 3 – 차트 개체 예

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Sales라는 테이블에 2022년의 주간 판매 합계가 포함된 데이터 집합.
- 3개 필드에 제공되는 트랜잭션 날짜: year, week 및 sales.

이 예에서는 차트 개체를 사용하여 첫 번째 예에서 end_of_week 계산에 해당하는 측정값을 만듭니다. 이 측정값은 makeweekdate() 함수를 사용하여 해당 주의 금요일 날짜를 MM/DD/YYYY 서식으로 반환합니다.

반환된 날짜가 금요일임을 증명하기 위해 요일을 반환하는 두 번째 측정값이 만들어집니다.

로드 스크립트

```
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
];
```

결과

다음과 같이 하십시오.

1. 데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.
   - transaction_year
   - transaction_week
2. 첫 번째 예의 end_of_week 필드와 동일한 계산을 수행하려면 다음 측정값을 만듭니다.
   `=makeweekdate(transaction_year,transaction_week,4)`
3. 각 트랜잭션의 요일을 계산하려면 다음 측정값을 만듭니다.
   `=weekday(makeweekdate(transaction_year,transaction_week,4))`

결과 테이블

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

transaction_ year	transaction_ week	=makeweekdate (transaction_ year,transaction_week,4)	=weekday(makeweekdate (transaction_year,transaction_ week,4))
2022	04	01/28/2022	Fri
2022	05	02/04/2022	Fri
2022	06	02/11/2022	Fri
2022	07	02/18/2022	Fri

makeweekdate() 함수를 사용하여 측정값으로 차트 개체에 end_of_week에 해당하는 필드가 만들어집니다. transaction_year 및 transaction_week 필드는 연도 및 주 인수로 전달됩니다. day 인수에는 값 4가 사용됩니다.

그런 다음 함수는 이러한 값을 결합하고 날짜 필드로 변환하여 DateFormat 시스템 변수의 형식으로 결과를 반환합니다.

makeweekdate() 함수와 해당 인수도 weekday() 함수로 래핑되어 첫 번째 예의 week_day 필드와 동일한 계산을 반환합니다. 위의 표에서 볼 수 있듯이 오른쪽의 마지막 열은 이러한 날짜가 금요일에 발생했음을 보여줍니다.

예 4 – 시나리오

로드 스크립트 및 차트 표현식

개요

이 예에서는 2022년의 모든 금요일을 포함하는 날짜 목록을 만듭니다.

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트

```
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;
```

결과

결과 테이블

date	weekday
01/07/2022	Fri
01/14/2022	Fri
01/21/2022	Fri
01/28/2022	Fri
02/04/2022	Fri
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
+ 27 추가 행	

`makeweekdate()` 함수는 2022년에서 각각의 금요일을 찾습니다. 주 매개 변수가 -2이면 날짜가 누락되지 않습니다. 마지막으로 선행 LOAD는 명확성을 위해 추가 `weekday` 필드를 만들어 각 `date` 값이 금요일임을 보여줍니다.

minute

이 함수는 **expression**의 분위수가 표준 숫자 해석에 따라 시간으로 해석될 경우 분을 나타내는 정수를 반환합니다.

구문:

```
minute(expression)
```

반환 데이터 유형: 정수

사용 시기

minute() 함수는 집계를 분 단위로 비교하려는 경우에 유용합니다. 예를 들어 분 단위로 활동 횟수 분포를 보려면 이 함수를 사용할 수 있습니다.

이러한 차원은 마스터 캘린더 테이블에 필드를 만드는 함수를 사용하여 로드 스크립트에서 만들 수 있습니다. 또는 차트에서 계산 차원으로 직접 사용할 수 있습니다.

함수 예

예	결과
minute ('09:14:36')	14을 반환합니다.
minute ('0.5555')	0.5555 = 13:19:55이므로, 19를 반환합니다.

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

예 1 – 변수(스크립트)

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Transactions라는 테이블에 로드되는 타임스탬프별 트랜잭션을 포함하는 데이터 집합.
- 기본 TimeStamp 시스템 변수(M/D/YYYY h:mm:ss[.fff] TT)가 사용됩니다.
- 트랜잭션이 발생하는 시점을 계산하는 필드(minute) 만들기.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- timestamp
- minute

결과 테이블

타임스탬프	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

타임스탬프	minute
2022-01-05 19:34:46	34
2022-01-06 08:49:09	49
2022-01-06 11:29:38	29

minute 필드의 값은 minute() 함수를 사용하고 선행 LOAD 문의 표현식으로 timestamp를 전달하여 만들어집니다.

예 2 – 차트 개체(차트)

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 첫 번째 예와 동일한 데이터 집합 및 시나리오.
- 기본 TimeStamp 시스템 변수(M/D/YYYY h:mm:ss[.fff] TT)가 사용됩니다.

그러나 이 예에서는 변경되지 않은 데이터 집합이 응용 프로그램에 로드됩니다. minute 값은 차트 개체의 측정값을 통해 계산됩니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다. timestamp.

다음 측정값을 만듭니다.

```
=minute(timestamp)
```

결과 테이블

타임스탬프	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

minute의 값은 minute() 함수를 사용하고 차트 개체에 대한 측정값의 표현식으로 timestamp를 전달하여 만들어집니다.

예 3 – 시나리오

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 개찰구 입장을 나타내기 위해 생성되는 타임스탬프 데이터 집합.
- Ticket_Barrier_Tracker라는 테이블에 로드되는 각 timestamp 및 해당 id 정보.
- 기본 TimeStamp 시스템 변수(M/D/YYYY h:mm:ss[.fff] TT)가 사용됩니다.

사용자는 개찰구 입장 수를 분 단위로 표시하는 차트 개체를 원합니다.

로드 스크립트

```
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;
```

결과

다음과 같이 하십시오.

1. 데이터를 로드하고 시트를 엽니다. 새 테이블을 만듭니다.
2. 다음 표현식을 사용하여 계산된 차원을 만듭니다.
   `=minute(timestamp)`
3. 총 입장 수를 계산하려면 다음 집계 측정값을 추가합니다.
   `=count(id)`
4. 측정값의 **숫자 형식**을 **화폐**로 설정합니다.

결과 테이블

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

minute(timestamp)	=count(id)
13	184
14	159
15	161
16	152
17	160
18	176
19	164
20	170
21	170
22	142
23	145
24	155
+ 35 추가 행	

month

이 함수는 환경 변수 **MonthNames**로 정의되고 1~12 사이의 정수인 월 이름이 포함된 이중 값을 반환합니다. 월은 표준 숫자 해석에 따라 표현식의 날짜 해석을 통해 계산됩니다.

이 함수는 특정 날짜에 대한 MonthName 시스템 변수 형식으로 월 이름을 반환합니다. 일반적으로 마스터 캘린더에서 일 필드를 차원으로 만드는 데 사용됩니다.

구문:

```
month(expression)
```

반환 데이터 유형: 정수

함수 예

예	결과
month(2012-10-12)	Oct를 반환합니다.
month(35648)	35648 = 1997-08-06이므로 Aug를 반환합니다.

예 1 – DateFormat 데이터 집합(스크립트)

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Master_Calendar라는 날짜의 데이터 집합. DateFormat 시스템 변수는 DD/MM/YYYY로 설정됩니다.
- month() 함수를 사용하여 month_name라는 추가 필드를 만드는 선행 LOAD입니다.
- 전체 날짜를 표현하는 date() 함수를 사용하는 long_date라는 추가 필드입니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- long_date
- month_name

결과 테이블

날짜	long_date	month_name
03/01/2022	03-January- 2022	1월
03/02/2022	03-February- 2022	2월
03/03/2022	03-March- 2022	3월
03/04/2022	03-April- 2022	4월
03/05/2022	03-May- 2022	5월

날짜	long_date	month_name
03/06/2022	03-June- 2022	6월
03/07/2022	03-July- 2022	Jul
03/08/2022	03-August- 2022	Aug
03/09/2022	03-September- 2022	Sep
03/10/2022	03-October- 2022	Oct
03/11/2022	03-November- 2022	Nov

월 이름은 스크립트의 month() 함수에 의해 올바르게 평가됩니다.

예 2 – ANSI 날짜(스크립트)

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Master_Calendar라는 날짜의 데이터 집합. DateFormat 시스템 변수 DD/MM/YYYY가 사용됩니다. 그러나 데이터 집합에 포함된 날짜는 ANSI 표준 날짜 서식입니다.
- month() 함수를 사용하여 month_name라는 추가 필드를 만드는 선행 LOAD입니다.
- 전체 날짜를 표현하는 date() 함수를 사용하는 long_date라는 추가 필드입니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- long_date
- month_name

결과 테이블

날짜	long_date	month_name
03/11/2022	11-March- 2022	11
03/12/2022	12-March- 2022	12
03/13/2022	13-March- 2022	13
03/14/2022	14-March- 2022	14
03/15/2022	15-March- 2022	15
03/16/2022	16-March- 2022	16
03/17/2022	17-March- 2022	17
03/18/2022	18-March- 2022	18
03/19/2022	19-March- 2022	19
03/20/2022	20-March- 2022	20
03/21/2022	21-March- 2022	21

월 이름은 스크립트의 month() 함수에 의해 올바르게 평가됩니다.

예 3 – 형식이 지정되지 않은 날짜(스크립트)

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Master_Calendar라는 날짜의 데이터 집합. DateFormat 시스템 변수 DD/MM/YYYY가 사용됩니다.
- month() 함수를 사용하여 month_name라는 추가 필드를 만드는 선행 LOAD입니다.
- unformatted_date라는 형식이 지정되지 않은 원래 날짜입니다.
- 전체 날짜를 표현하는 date() 함수를 사용하는 long_date라는 추가 필드입니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- unformatted_date
- long_date
- month_name

결과 테이블

unformatted_date	long_date	month_name
44868	03-January- 2022	1월
44898	03-February- 2022	2월
44928	03-March- 2022	3월
44958	03-April- 2022	4월
44988	03-May- 2022	5월
45018	03-June- 2022	6월
45048	03-July- 2022	Jul
45078	03-August- 2022	Aug
45008	03-September- 2022	Sep

unformatted_date	long_date	month_name
45038	03-October- 2022	Oct
45068	03-November- 2022	Nov

월 이름은 스크립트의 `month()` 함수에 의해 올바르게 평가됩니다.

예 4 - 만료 월 계산

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 3월에 이루어진 주문 데이터 집합(`Subscriptions`라고 함). 테이블에는 세 개의 필드가 있습니다.
  - id
  - order_date
  - amount

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다. `order_date`.

주문이 만료되는 월을 계산하려면 다음 측정값 =month(order_date+180)을 만듭니다.

결과 테이블

order_date	=month(order_date+180)
03/01/2022	Jul
03/02/2022	Aug
03/03/2022	Aug
03/04/2022	Sep
03/05/2022	Oct
03/06/2022	Nov
03/07/2022	Dec
03/08/2022	1월
03/09/2022	3월
03/10/2022	4월
03/11/2022	5월

month() 함수는 3월 11일에 이루어진 주문이 7월에 만료될 것이라고 올바르게 결정합니다.

monthend

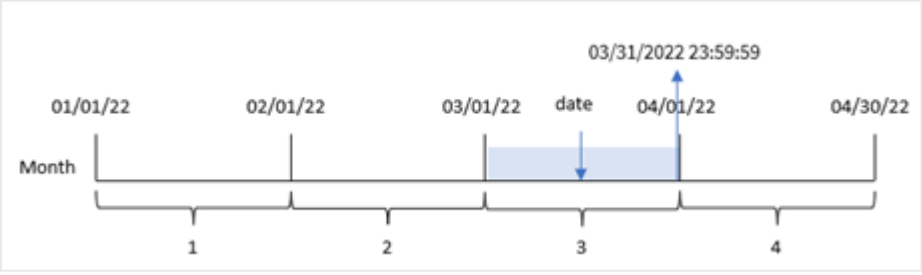
이 함수는 date를 포함하는 월의 마지막 날의 마지막 밀리초의 타임스탬프에 해당하는 값을 반환합니다. 기본 출력 형식은 스크립트에 설정된 DateFormat입니다.

구문:

```
MonthEnd(date[, period_no])
```

즉, monthend() 함수는 이 날짜가 속하는 월을 확인합니다. 그런 다음 해당 월의 마지막 밀리초에 대한 타임스탬프를 날짜 서식으로 반환합니다.

monthend 함수 다이어그램.



사용 시기

monthend() 함수는 아직 발생하지 않은 월의 부분을 사용하도록 계산하려는 경우 표현식의 일부로 사용됩니다. 예를 들어 해당 월에 아직 발생하지 않은 총 이자를 계산하려는 경우에 사용할 수 있습니다.

반환 데이터 유형: dual

인수

인수	설명
date	평가할 날짜 또는 타임스탬프입니다.
period_no	**period_no**는 정수이며, 0이거나 생략되는 경우는 **date**를 포함하는 월을 나타냅니다. **period_no**가 음수 값일 경우 이전 달, 양수 값일 경우 다음 달을 나타냅니다.

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

함수 예

예	결과
`monthend('02/19/2012')`	02/29/2012 23:59:59를 반환합니다.
`monthend('02/19/2001', -1)`	01/31/2001 23:59:59를 반환합니다.

예 1 – 기본 예

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 'Transactions'라는 테이블에 로드되는 2022년 트랜잭션 집합을 포함하는 데이터 집합.
- DateFormat 시스템 변수 MM/DD/YYYY 형식의 날짜 필드.
- 다음을 포함하는 선행 LOAD 문:
    - 'end_of_month' 필드로 설정되는 monthend() 함수.
    - 'end_of_month_timestamp' 필드로 설정되는 timestamp 함수.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- id
- date
- end_of_month
- end_of_month_timestamp

결과 테이블

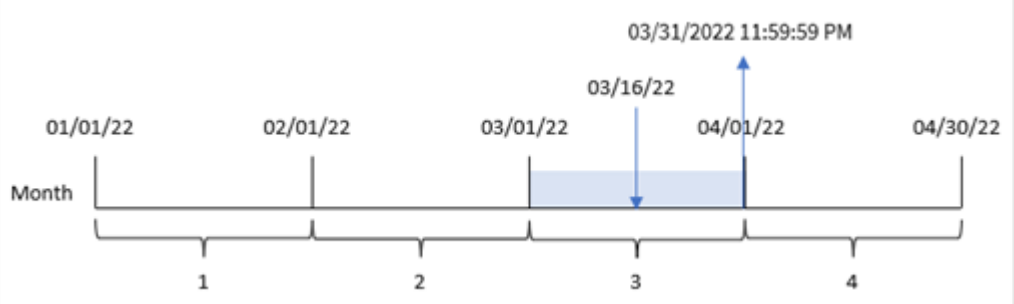
id	date	end_of_month	end_of_month_timestamp
8188	1/7/2022	01/31/2022	1/31/2022 11:59:59 PM

id	date	end_of_month	end_of_month_timestamp
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
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

'end_of_month' 필드는 monthend() 함수를 사용하고 날짜 필드를 함수의 인수로 전달하여 선행 LOAD 문에서 만들어집니다.

monthend() 함수는 날짜 값이 속하는 월을 식별하고 해당 월의 마지막 밀리초에 대한 타임스탬프를 반환합니다.

3월이 선택된 monthend 함수의 다이어그램.



트랜잭션 8192는 3월 16일에 발생했습니다. monthend() 함수는 해당 월의 마지막 밀리초, 즉 3월 31일 오후 11:59:59를 반환합니다.

예 2 – period_no

로드 스크립트 및 결과

개요

첫 번째 예와 동일한 데이터 집합 및 시나리오가 사용됩니다.

이 예에서 작업은 트랜잭션이 발생하기 전 월의 끝에 대한 타임스탬프를 반환하는 필드 'previous_month_end'를 만드는 것입니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

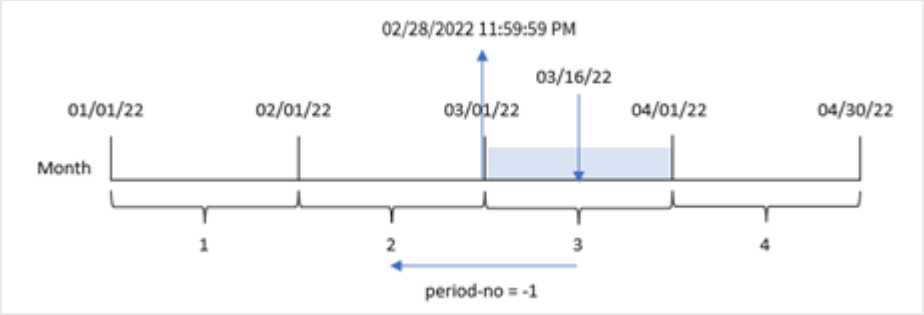
- id
- date
- previous_month_end
- previous_month_end_timestamp

결과 테이블

id	date	previous_month_end	previous_month_end_timestamp
8188	1/7/2022	12/31/2021	12/31/2021 11:59:59 PM
8189	1/19/2022	12/31/2021	12/31/2021 11:59:59 PM
8190	2/5/2022	01/31/2022	1/31/2022 11:59:59 PM
8191	2/28/2022	01/31/2022	1/31/2022 11:59:59 PM
8192	3/16/2022	02/28/2022	2/28/2022 11:59:59 PM
8193	4/1/2022	03/31/2022	3/31/2022 11:59:59 PM
8194	5/7/2022	04/30/2022	4/30/2022 11:59:59 PM
8195	5/16/2022	04/30/2022	4/30/2022 11:59:59 PM
8196	6/15/2022	05/31/2022	5/31/2022 11:59:59 PM
8197	6/26/2022	05/31/2022	5/31/2022 11:59:59 PM
8198	7/9/2022	06/30/2022	6/30/2022 11:59:59 PM
8199	7/22/2022	06/30/2022	6/30/2022 11:59:59 PM
8200	7/23/2022	06/30/2022	6/30/2022 11:59:59 PM
8201	7/27/2022	06/30/2022	6/30/2022 11:59:59 PM
8202	8/2/2022	07/31/2022	7/31/2022 11:59:59 PM
8203	8/8/2022	07/31/2022	7/31/2022 11:59:59 PM
8204	8/19/2022	07/31/2022	7/31/2022 11:59:59 PM
8205	9/26/2022	08/31/2022	8/31/2022 11:59:59 PM
8206	10/14/2022	09/30/2022	9/30/2022 11:59:59 PM
8207	10/29/2022	09/30/2022	9/30/2022 11:59:59 PM

monthend() 함수는 period_no -1을 오프셋 인수로 사용하여 트랜잭션이 발생한 월을 먼저 식별합니다. 그런 다음 한 달 전으로 시프트하고 해당 월의 마지막 밀리초를 식별합니다.

period_no 변수를 사용하는 monthend 함수의 다이어그램.



트랜잭션 8192는 3월 16일에 발생했습니다. monthend() 함수는 트랜잭션이 발생하기 전 월이 2월임을 식별합니다. 그런 다음 해당 월의 마지막 밀리초인 2월 28일 오후 11:59:59를 반환합니다.

예 3 – 차트 예

로드 스크립트 및 차트 표현식

개요

첫 번째 예와 동일한 데이터 집합 및 시나리오가 사용됩니다.

이 예에서 데이터 집합은 변경되지 않고 앱에 로드됩니다. 트랜잭션이 앱 차트에서 측정값으로 발생한 월의 끝에 대한 타임스탬프를 반환하는 계산을 만드는 작업입니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- id

트랜잭션이 발생한 월의 끝 날짜를 계산하려면 다음 측정값을 만듭니다.

- =monthend(date)
- =timestamp(monthend(date))

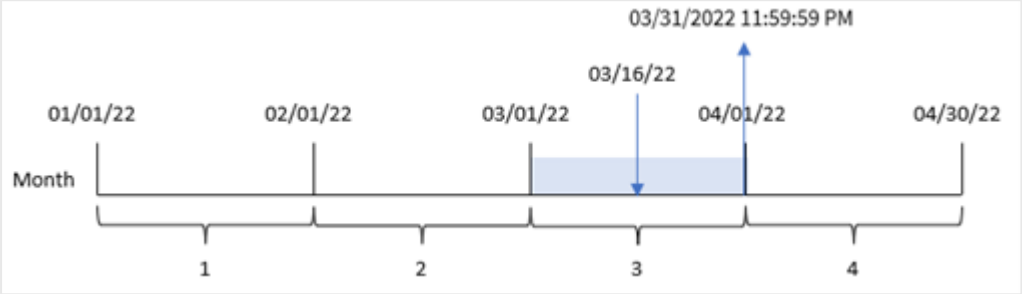
결과 테이블

id	date	=monthend(date)	=timestamp(monthend(date))
8188	10/14/2022	10/31/2022	10/31/2022 11:59:59 PM
8189	10/29/2022	10/31/2022	10/31/2022 11:59:59 PM
8190	9/26/2022	09/30/2022	9/30/2022 11:59:59 PM
8191	8/2/2022	08/31/2022	8/31/2022 11:59:59 PM
8192	8/8/2022	08/31/2022	8/31/2022 11:59:59 PM
8193	8/19/2022	08/31/2022	8/31/2022 11:59:59 PM
8194	7/9/2022	07/31/2022	7/31/2022 11:59:59 PM
8195	7/22/2022	07/31/2022	7/31/2022 11:59:59 PM
8196	7/23/2022	07/31/2022	7/31/2022 11:59:59 PM
8197	7/27/2022	07/31/2022	7/31/2022 11:59:59 PM
8198	6/15/2022	06/30/2022	6/30/2022 11:59:59 PM
8199	6/26/2022	06/30/2022	6/30/2022 11:59:59 PM
8200	5/7/2022	05/31/2022	5/31/2022 11:59:59 PM
8201	5/16/2022	05/31/2022	5/31/2022 11:59:59 PM
8202	4/1/2022	04/30/2022	4/30/2022 11:59:59 PM
8203	3/16/2022	03/31/2022	3/31/2022 11:59:59 PM
8204	2/5/2022	02/28/2022	2/28/2022 11:59:59 PM
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

'end_of_month' 측정값은 monthend() 함수를 사용하고 날짜 필드를 함수의 인수로 전달하여 차트에서 만들어집니다.

monthend() 함수는 날짜 값이 속하는 월을 식별하고 해당 월의 마지막 밀리초에 대한 타임스탬프를 반환합니다.

period_no 변수를 사용하는 monthend 함수의 다이어그램.



트랜잭션 8192는 3월 16일에 발생했습니다. monthend() 함수는 해당 월의 마지막 밀리초, 즉 3월 31일 오후 11:59:59를 반환합니다.

예 4 – 시나리오

로드 스크립트 및 결과

개요

이 예에서 데이터 집합은 'Employee_Expenses'라는 테이블에 로드됩니다. 테이블에는 다음 필드가 포함됩니다.

- 직원 ID
- 직원 이름
- 각 직원의 평균 일일 비용 청구

최종 사용자는 직원 ID 및 직원 이름별로 해당 월에 남은 기간 동안 예상되는 비용 청구를 표시하는 차트를 원합니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- employee_id
- employee_name

누적 이자를 계산하려면 다음 측정값을 만듭니다.

```
=floor(monthend(today(1),0)-today(1))*avg_daily_claim
```

이 측정값은 동적이며 데이터를 로드하는 날짜에 따라 다른 테이블 결과를 생성합니다.

측정값의 **숫자 형식**을 **화폐**로 설정합니다.

결과 테이블

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

monthend() 함수는 오늘 날짜를 유일한 인수로 사용하여 현재 월의 끝 날짜를 반환합니다. 이 표현식은 월 끝 날짜에서 오늘 날짜를 빼서 이번 달에 남아 있는 일 수를 반환합니다.

그런 다음 이 값에 각 직원의 평균 일일 비용 청구를 곱하여 각 직원이 남은 월에 예상되는 청구 금액을 계산합니다.

monthname

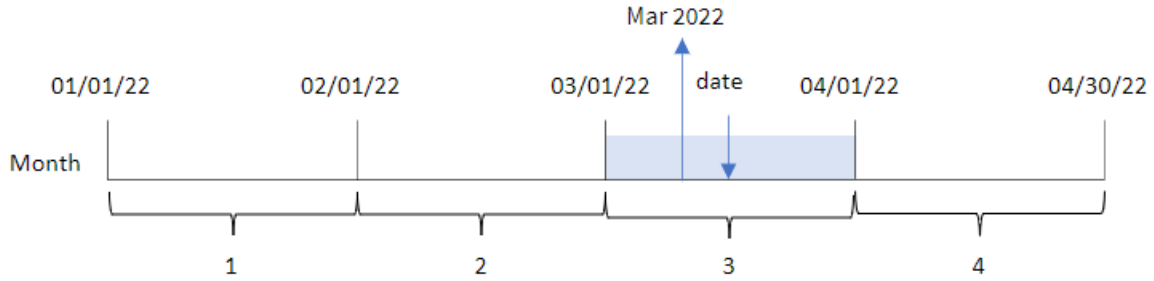
이 함수는 해당 월의 첫 번째 날의 첫 번째 밀리초의 타임스탬프에 해당하는 기본 숫자 값으로 월(**MonthNames** 스크립트 변수에 따라 서식 지정) 및 연도를 보여주는 표시 값을 반환합니다.

구문:

```
MonthName(date[, period_no])
```

반환 데이터 유형: dual

monthname 함수의 다이어그램



인수

인수	설명
date	평가할 날짜 또는 타임스탬프입니다.
period_no	**period_no**는 정수이며, 0이거나 생략되는 경우는 **date**를 포함하는 월을 나타냅니다. **period_no**가 음수 값일 경우 이전 달, 양수 값일 경우 다음 달을 나타냅니다.

함수 예

예	결과
`monthname('10/19/2013')`	Oct 2013를 반환합니다.
`monthname('10/19/2013', -1)`	Sep 2013를 반환합니다.

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 `SET DateFormat` 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

예 1 – 기본 예

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Transactions라는 테이블에 로드되는 2022년 트랜잭션 집합이 포함된 데이터 집합.
- DateFormat 시스템 변수(MM/DD/YYYY) 서식으로 제공된 날짜 필드.
- 트랜잭션이 발생한 월을 반환하는 필드 transaction_month 만들기.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

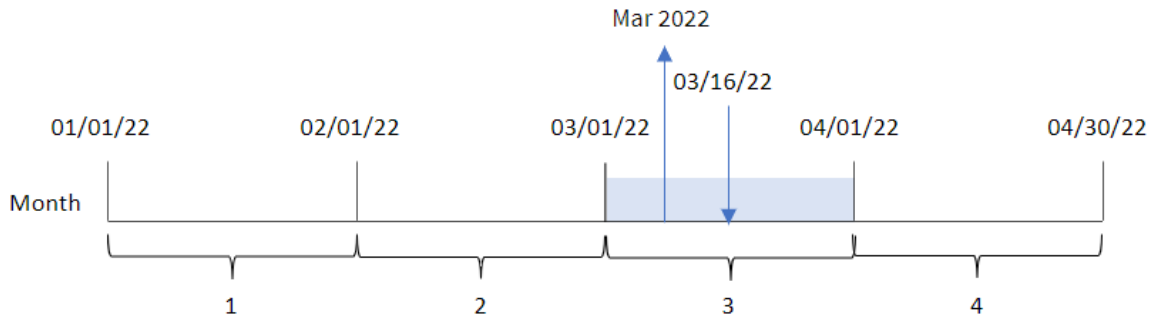
- date
- transaction_month

결과 테이블

date	transaction_month
1/7/2022	Jan 2022
1/19/2022	Jan 2022
2/5/2022	Feb 2022
2/28/2022	Feb 2022
3/16/2022	Mar 2022
4/1/2022	Apr 2022
5/7/2022	May 2022
5/16/2022	May 2022
6/15/2022	Jun 2022
6/26/2022	Jun 2022
7/9/2022	Jul 2022
7/22/2022	Jul 2022
7/23/2022	Jul 2022
7/27/2022	Jul 2022
8/2/2022	Aug 2022
8/8/2022	Aug 2022
8/19/2022	Aug 2022
9/26/2022	Sep 2022
10/14/2022	Oct 2022
10/29/2022	Oct 2022

transaction_month 필드는 monthname() 함수를 사용하고 date 필드를 함수의 인수로 전달하여 선행 LOAD 문에서 만들어집니다.

monthname 함수의 다이어그램, 기본 예



monthname() 함수는 트랜잭션 8192가 2022년 3월에 발생했음을 식별하고 MonthNames 시스템 변수를 사용하여 이 값을 반환합니다.

예 2 – period_no

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 첫 번째 예와 동일한 인라인 데이터 집합 및 시나리오.
- 트랜잭션이 발생하기 전 월말의 타임스탬프를 반환하는 필드 transaction_previous_month 만들기.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- `date`
- `transaction_previous_month`

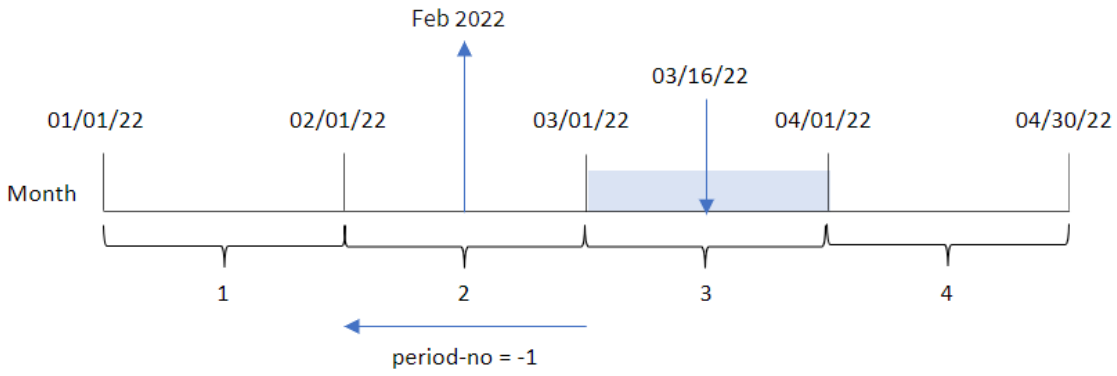
결과 테이블

date	transaction_previous_month
1/7/2022	Dec 2021
1/19/2022	Dec 2021
2/5/2022	Jan 2022
2/28/2022	Jan 2022
3/16/2022	Feb 2022
4/1/2022	Mar 2022
5/7/2022	Apr 2022
5/16/2022	Apr 2022
6/15/2022	May 2022
6/26/2022	May 2022
7/9/2022	Jun 2022
7/22/2022	Jun 2022
7/23/2022	Jun 2022
7/27/2022	Jun 2022
8/2/2022	Jul 2022
8/8/2022	Jul 2022
8/19/2022	Jul 2022

date	transaction_previous_month
9/26/2022	Aug 2022
10/14/2022	Sep 2022
10/29/2022	Sep 2022

이 경우 monthname() 함수에서 period_no -1을 오프셋 인수로 사용했으므로 이 함수는 먼저 트랜잭션이 발생한 월을 식별합니다. 그런 다음 한 달 전으로 이동하고 월 이름과 연도를 반환합니다.

monthname 함수의 다이어그램, period_no 예



트랜잭션 8192는 3월 16일에 발생했습니다. monthname() 함수는 트랜잭션이 발생하기 전 월이 2월임을 식별하고 2022년과 함께 MonthNames 시스템 변수 서식으로 월을 반환합니다.

예 3 – 차트 개체 예

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

이 로드 스크립트에는 첫 번째 예와 동일한 인라인 데이터 집합 및 시나리오가 포함되어 있습니다. 그러나 이 예에서는 변경되지 않은 데이터 집합이 응용 프로그램에 로드됩니다. 트랜잭션이 발생한 월말의 타임스탬프를 반환하는 계산은 응용 프로그램의 차트 개체에서 측정값으로 만들어집니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다.`date`.

다음 측정값을 만듭니다.

```
=monthname(date)
```

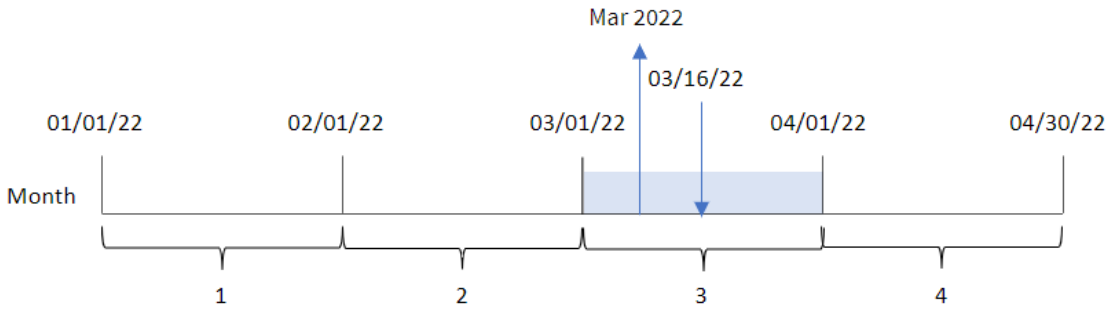
결과 테이블

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

date	=monthname(date)
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

month_name 측정값은 monthname() 함수를 사용하고 date 필드를 함수의 인수로 전달하여 차트 개체에서 만들어집니다.

monthname 함수의 다이어그램, 차트 개체 예



monthname() 함수는 트랜잭션 8192가 2022년 3월에 발생했음을 식별하고 MonthNames 시스템 변수를 사용하여 이 값을 반환합니다.

monthsend

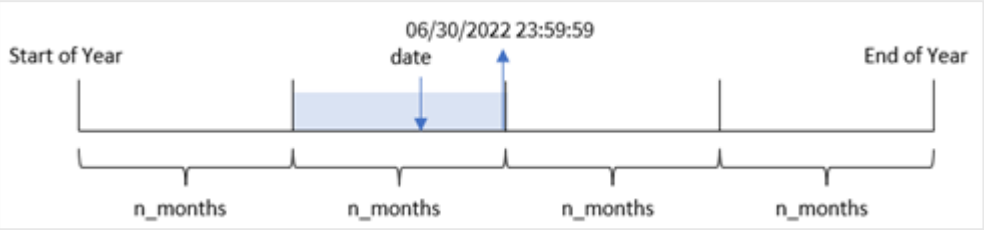
이 함수는 기준 날짜를 포함하는 월, 2개월, 분기, 4개월 기간 또는 6개월의 마지막 밀리초의 타임스탬프에 해당하는 값을 반환합니다. 또한 이전 기간 또는 다음 기간의 끝에 대한 타임스탬프를 찾을 수도 있습니다. 기본 출력 형식은 스크립트에 설정된 DateFormat입니다.

구문:

```
MonthsEnd(n_months, date[, period_no [, first_month_of_year]])
```

반환 데이터 유형: dual

monthsend 함수 다이어그램.



인수

인수	설명
n_months	기간을 정의하는 개월 수입니다. 정수 또는 정수로 처리되는 표현식으로, 1(inmonth() 함수와 동일), 2(2개월), 3(inquarter() 함수와 동일), 4(4개월 기간) 또는 6(6개월) 중 하나여야 합니다.
date	평가할 날짜 또는 타임스탬프입니다.
period_no	기간은 **period_no**, 정수 또는 정수로 처리되는 표현식으로 오프셋을 지정할 수 있습니다. 값 0은 **base_date**를 포함하는 기간을 나타냅니다. **period_no**가 음수 값일 경우 이전 기간, 양수 값일 경우 다음 기간을 나타냅니다.
first_month_of_year	1월에 시작되지 않는 (회계)연도를 사용하려는 경우 **first_month_of_year**에 2와 12 사이의 값을 지정하십시오.

monthsend() 함수는 제공된 n_months 인수를 기반으로 연도를 세그먼트로 나눕니다. 그런 다음 제공된 각 날짜가 속하는 세그먼트를 평가하고 해당 세그먼트의 마지막 밀리초를 날짜 서식으로 반환합니다. 이 함수는 연도의 첫 번째 달을 재정의할 뿐만 아니라 이전 또는 다음 세그먼트에서 끝 타임스탬프를 반환할 수 있습니다.

함수에서 n_month 인수로 사용할 수 있는 연도의 세그먼트는 다음과 같습니다.

n_month 인수

기간	개월 수
1개월	1
2개월	2
분기	3
4개월	4
6개월	6

사용 시기

monthsend() 함수는 사용자가 지금까지 경과된 월의 부분을 사용하여 계산하려고 할 때 표현식의 일부로 사용됩니다. 사용자는 변수를 사용하여 기간을 선택할 수 있습니다. 예를 들어, monthsend()는 사용자가 월, 분기 또는 6개월 동안 아직 발생하지 않은 총 이자를 계산할 수 있도록 하는 입력 변수를 제공할 수 있습니다.

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

함수 예

예	결과
monthsend(4, '07/19/2013')	08/31/2013를 반환합니다.
monthsend(4, '10/19/2013', -1)	08/31/2013를 반환합니다.
monthsend(4, '10/19/2013', 0, 2)	01/31/2014을 반환합니다. 연도가 2월부터 시작되기 때문입니다.

예 1 - 기본 예

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 'Transactions'라는 테이블에 로드되는 2022년 트랜잭션 집합을 포함하는 데이터 집합.
- 날짜 필드가 DateFormat 시스템 변수 (MM/DD/YYYY) 형식으로 제공됩니다.
- 다음을 포함하는 선행 LOAD 문:
    - 'bi_monthly_end' 필드로 설정되는 monthsend 함수. 이는 트랜잭션을 2개월 세그먼트로 그룹화합니다.
    - timestamp 함수는 각 트랜잭션에 대한 세그먼트의 시작 타임스탬프를 반환합니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- id
- date
- bi_monthly_end
- bi_monthly_end_timestamp

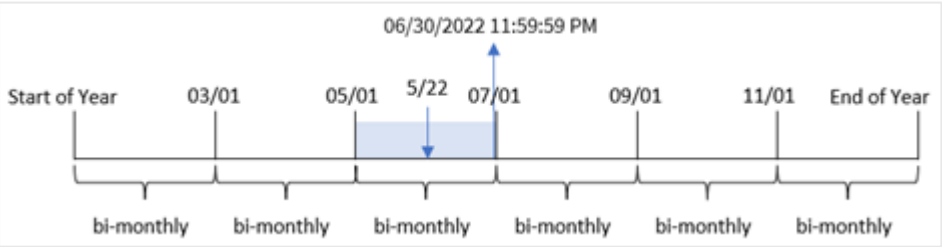
결과 테이블

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

id	date	bi_monthly_end	bi_monthly_end_timestamp
8192	3/16/2022	04/30/2022	4/30/2022 11:59:59 PM
8193	4/1/2022	04/30/2022	4/30/2022 11:59:59 PM
8194	5/7/2022	06/30/2022	6/30/2022 11:59:59 PM
8195	5/22/2022	06/30/2022	6/30/2022 11:59:59 PM
8196	6/15/2022	06/30/2022	6/30/2022 11:59:59 PM
8197	6/26/2022	06/30/2022	6/30/2022 11:59:59 PM
8198	7/9/2022	08/31/2022	8/31/2022 11:59:59 PM
8199	7/22/2022	08/31/2022	8/31/2022 11:59:59 PM
8200	7/23/2022	08/31/2022	8/31/2022 11:59:59 PM
8201	7/27/2022	08/31/2022	8/31/2022 11:59:59 PM
8202	8/2/2022	08/31/2022	8/31/2022 11:59:59 PM
8203	8/8/2022	08/31/2022	8/31/2022 11:59:59 PM
8204	8/19/2022	08/31/2022	8/31/2022 11:59:59 PM
8205	9/26/2022	10/31/2022	10/31/2022 11:59:59 PM
8206	10/14/2022	10/31/2022	10/31/2022 11:59:59 PM
8207	10/29/2022	10/31/2022	10/31/2022 11:59:59 PM

'bi_monthly_end' 필드는 monthsend() 함수를 사용하여 선행 LOAD 문에서 만들어집니다. 제공된 첫 번째 인수는 2이며, 연도를 2개월 세그먼트로 나눕니다. 두 번째 인수는 평가 중인 필드를 식별합니다.

2개월 세그먼트가 있는 monthsend 함수의 다이어그램.



트랜잭션 8195는 5월 22일에 발생합니다. monthsend() 함수는 처음에 연도를 2개월 세그먼트로 나눕니다. 트랜잭션 8195는 5월 1일에서 6월 30일 사이의 세그먼트에 속합니다. 결과적으로 이 함수는 이 세그먼트의 마지막 밀리초, 즉 2022년 6월 30일 오후 11:59:59를 반환합니다.

예 2 – period_no

로드 스크립트 및 결과

개요

첫 번째 예와 동일한 데이터 집합 및 시나리오가 사용됩니다.

이 예에서 작업은 트랜잭션이 발생하기 전에 2개월 세그먼트의 첫 번째 밀리초를 반환하는 필드 'prev_bi_monthly_end'를 만드는 것입니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

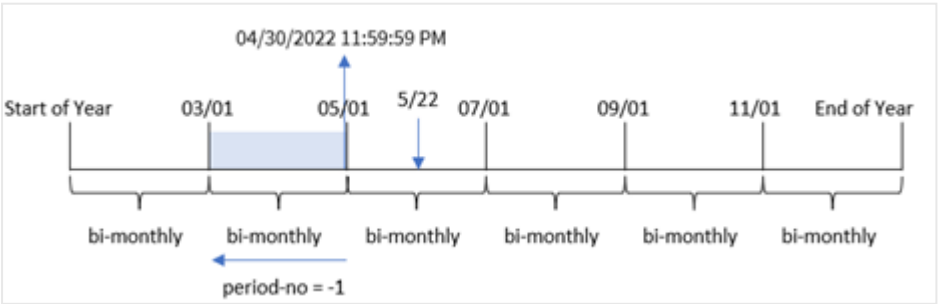
- id
- date
- prev_bi_monthly_end
- prev_bi_monthly_end_timestamp

결과 테이블

id	date	prev_bi_monthly_end	prev_bi_monthly_end_timestamp
8188	1/7/2022	12/31/2021	12/31/2021 11:59:59 PM
8189	1/19/2022	12/31/2021	12/31/2021 11:59:59 PM
8190	2/5/2022	12/31/2021	12/31/2021 11:59:59 PM
8191	2/28/2022	12/31/2021	12/31/2021 11:59:59 PM
8192	3/16/2022	02/28/2022	2/28/2022 11:59:59 PM
8193	4/1/2022	02/28/2022	2/28/2022 11:59:59 PM
8194	5/7/2022	04/30/2022	4/30/2022 11:59:59 PM
8195	5/22/2022	04/30/2022	4/30/2022 11:59:59 PM
8196	6/15/2022	04/30/2022	4/30/2022 11:59:59 PM
8197	6/26/2022	04/30/2022	4/30/2022 11:59:59 PM
8198	7/9/2022	06/30/2022	6/30/2022 11:59:59 PM
8199	7/22/2022	06/30/2022	6/30/2022 11:59:59 PM
8200	7/23/2022	06/30/2022	6/30/2022 11:59:59 PM
8201	7/27/2022	06/30/2022	6/30/2022 11:59:59 PM
8202	8/2/2022	06/30/2022	6/30/2022 11:59:59 PM
8203	8/8/2022	06/30/2022	6/30/2022 11:59:59 PM
8204	8/19/2022	06/30/2022	6/30/2022 11:59:59 PM
8205	9/26/2022	08/31/2022	8/31/2022 11:59:59 PM
8206	10/14/2022	08/31/2022	8/31/2022 11:59:59 PM
8207	10/29/2022	08/31/2022	8/31/2022 11:59:59 PM

초기에 1년을 2개월 세그먼트로 나눈 후 `monthsend()` 함수의 `period_no` 인수로 `-1`을 사용하면 이 함수는 트랜잭션이 발생할 때 이전 2개월 세그먼트의 마지막 밀리초를 반환합니다.

이전 2개월 세그먼트를 반환하는 monthsend 함수의 다이어그램.



트랜잭션 8195는 5월과 6월 사이의 세그먼트에서 발생합니다. 결과적으로 이전 2개월 세그먼트는 3월 1일과 4월 30일 사이였으므로 이 함수는 이 세그먼트의 마지막 밀리초, 즉 2022년 4월 30일 오후 11:59:59를 반환합니다.

예 3 – first_month_of_year

로드 스크립트 및 결과

개요

첫 번째 예와 동일한 데이터 집합 및 시나리오가 사용됩니다.

이 예에서 조직 정책은 4월이 회계 연도의 첫 번째 달이 되도록 하는 것입니다.

트랜잭션을 2개월 세그먼트로 그룹화하고 각 트랜잭션에 대한 세그먼트의 마지막 밀리초 타임스탬프를 반환하는 필드 'bi_monthly_end'를 만듭니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

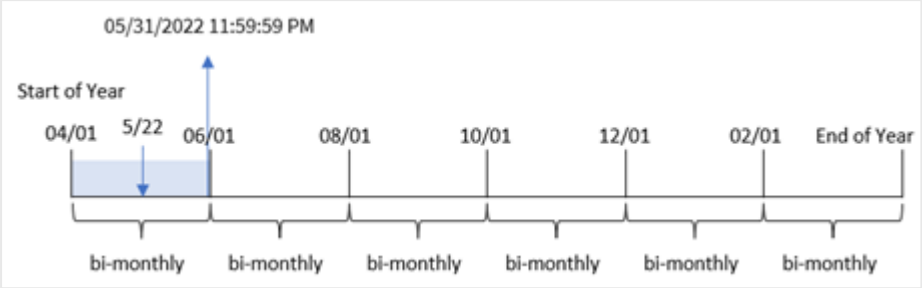
- id
- date
- bi_monthly_end
- bi_monthly_end_timestamp

결과 테이블

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

monthsend() 함수의 first_month_of_year 인수로 4를 사용하면 이 함수는 연도를 4월 1일에 시작합니다. 그런 다음 연도를 2개월 세그먼트로 나눕니다. 4월-5월, 6월-7월, 8월-9월, 10월-11월, 12월-1월, 2월-3월

연도의 첫 번째 달을 4월로 설정한 monthsend 함수의 다이어그램



트랜잭션 8195는 5월 22일에 발생했으며 4월 1일에서 5월 31일 사이에 해당합니다. 결과적으로 이 함수는 이 세그먼트의 마지막 밀리초, 즉 2022년 5월 31일 오후 11:59:59를 반환합니다.

예 4 - 차트 개체 예

로드 스크립트 및 차트 표현식

개요

첫 번째 예와 동일한 데이터 집합 및 시나리오가 사용됩니다. 그러나 이 예에서 데이터 집합은 변경되지 않고 앱에 로드됩니다.

이 예에서 트랜잭션을 2개월 세그먼트로 그룹화하고 각 트랜잭션에 대한 세그먼트의 마지막 밀리초 타임스탬프를 앱의 차트 개체에 측정값으로 반환하는 계산을 만드는 작업입니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다.

```
date
```

트랜잭션이 발생한 2개월 세그먼트의 마지막 밀리초 타임스탬프를 가져오려면 다음 측정값을 만듭니다.

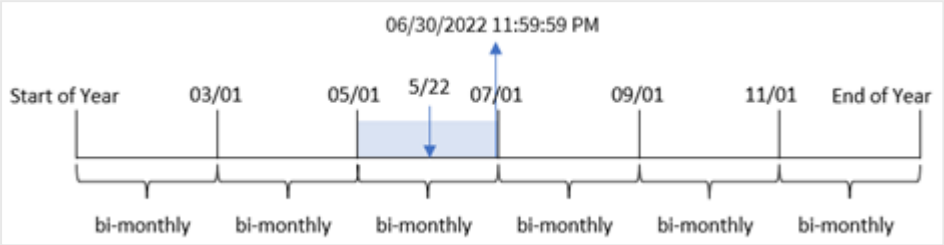
- `=monthsEnd(2,date)`
- `=timestamp(monthsend(2,date))`

결과 테이블

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
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

'bi_monthly_end' 필드는 monthsend() 함수를 사용하여 차트 개체에서 측정값으로 만들어집니다. 제공된 첫 번째 인수는 2이며, 연도를 2개월 세그먼트로 나눕니다. 두 번째 인수는 평가 중인 필드를 식별합니다.

2개월 세그먼트가 있는 monthsend 함수의 다이어그램.



트랜잭션 8195는 5월 22일에 발생합니다. monthsend() 함수는 처음에 연도를 2개월 세그먼트로 나눕니다. 트랜잭션 8195는 5월 1일에서 6월 30일 사이의 세그먼트에 속합니다. 결과적으로 이 함수는 이 세그먼트의 첫 번째 밀리초, 즉 2022년 6월 30일 오후 11:59:59를 반환합니다.

예 5 – 시나리오

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

이 예에서 데이터 집합은 'Employee_Expenses'라는 테이블에 로드됩니다. 테이블에는 다음 필드가 포함됩니다.

- 직원 ID
- 직원 이름
- 각 직원의 평균 일일 비용 청구

최종 사용자는 직원 ID 및 직원 이름별로 자신이 선택한 남은 기간 동안 예상되는 비용 청구를 표시하는 차트를 원합니다. 회계 연도는 1월에 시작됩니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 새 시트를 엽니다.

로드 스크립트 시작 시 변수 입력 컨트롤에 연결된 변수 vPeriod가 만들어집니다.

다음과 같이 하십시오.

1. 자산 패널에서 **사용자 지정 개체**를 클릭합니다.
2. **Qlik Dashboard 번들**을 선택하고 **변수 입력** 개체를 만듭니다.
3. 차트 개체의 제목을 입력합니다.
4. **변수**에서 이름으로 **vPeriod**를 선택하고 **드롭다운**으로 표시할 개체를 설정합니다.
5. **값**에서 **동적** 값을 클릭합니다. 다음을 입력합니다.
   ='1~month|2~bi-month|3~quarter|4~tertial|6~half-year'.

새 테이블과 다음 필드를 차원으로 만듭니다.

- employee_id
- employee_name

누적 이자를 계산하려면 다음 측정값을 만듭니다.

```
=floor(monthsend($(vPeriod),today(1))-today(1))*avg_daily_claim
```

이 측정값은 동적이며 데이터를 로드하는 날짜에 따라 다른 테이블 결과를 생성합니다.

측정값의 **숫자 형식**을 **화폐**로 설정합니다.

결과 테이블

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

monthsend() 함수는 사용자 입력을 첫 번째 인수로 사용하고 오늘 날짜를 두 번째 인수로 사용합니다. 이는 사용자가 선택한 기간의 끝 날짜를 반환합니다. 그런 다음, 이 표현식은 이 끝 날짜에서 오늘 날짜를 빼서 선택한 기간이 남아 있는 일 수를 반환합니다.

그런 다음 이 값에 각 직원의 평균 일일 비용 청구를 곱하여 각 직원이 이 기간의 남은 일수에 예상되는 청구 금액을 계산합니다.

monthsname

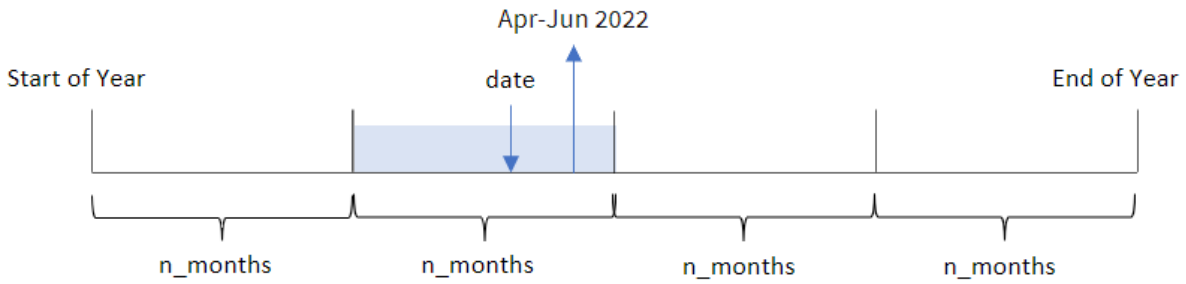
이 함수는 연도뿐 아니라 기간의 월 범위(**MonthNames** 스크립트 변수에 따라 서식 지정)를 나타내는 표시 값도 반환합니다. 기본 숫자 값은 기준일을 포함하는 월, 2개월, 분기, 4개월 기간 또는 6개월의 첫 번째 밀리초의 타임스탬프에 해당합니다.

구문:

```
MonthsName(n_months, date[, period_no[, first_month_of_year]])
```

반환 데이터 유형: dual

monthsname 함수의 다이어그램



monthsname() 함수는 제공된 n_months 인수를 기반으로 연도를 세그먼트로 나눕니다. 그런 다음 제공된 각 date가 속한 세그먼트를 평가하고 해당 세그먼트의 시작 및 종료 월 이름과 연도를 반환합니다. 또한 이 함수는 연도의 첫 번째 달을 재정의할 뿐만 아니라 앞 또는 뒤 세그먼트에서 이러한 경계를 반환하는 기능을 제공합니다.

함수에서 n_month 인수로 사용할 수 있는 연도의 다음 세그먼트:

가능한 n_month 인수

기간	개월 수
1개월	1
2개월	2
분기	3
4개월	4
6개월	6

인수

인수	설명
n_months	기간을 정의하는 개월 수입니다. 정수 또는 정수로 처리되는 표현식으로, 1(inmonth() 함수와 동일), 2(2개월), 3(inquarter() 함수와 동일), 4(4개월 기간) 또는 6(6개월) 중 하나여야 합니다.

인수	설명
date	평가할 날짜 또는 타임스탬프입니다.
period_no	기간은 **period_no**, 정수 또는 정수로 처리되는 표현식으로 오프셋을 지정할 수 있습니다. 값 0은 **base_date**를 포함하는 기간을 나타냅니다. **period_no**가 음수 값일 경우 이전 기간, 양수 값일 경우 다음 기간을 나타냅니다.
first_month_of_year	1월에 시작되지 않는 (회계)연도를 사용하려는 경우 **first_month_of_year**에 2와 12 사이의 값을 지정하십시오.

사용 시기

monthsname() 함수는 선택한 기간별로 집계를 비교하는 기능을 사용자에게 제공하려는 경우에 유용합니다. 예를 들어, 입력 변수를 제공하여 사용자가 월별, 분기별 또는 반기별 총 제품 판매를 볼 수 있도록 할 수 있습니다.

이러한 차원은 함수를 마스터 캘린더 테이블의 필드로 추가하여 로드 스크립트에서 만들거나 차트에서 계산된 차원으로 차원을 직접 만들 수 있습니다.

함수 예

예	결과
monthsname(4, '10/19/2013')	'Sep-Dec 2013'을 반환합니다. 이 예와 다른 예에서 **SET Monthnames** 문은 Jan;Feb;Mar 등으로 설정됩니다.
monthsname(4, '10/19/2013', -1)	'May-Aug 2013'을 반환합니다.
monthsname(4, '10/19/2013', 0, 2)	연도가 월 2에 시작하도록 지정되었으므로 'Oct-Jan 2014'를 반환합니다. 따라서 4개월 기간은 다음 해의 첫 번째 달에 끝납니다.

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

예 1 – 기본 예

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Transactions라는 테이블에 로드되는 2022년 트랜잭션 집합이 포함된 데이터 집합.
- DateFormat 시스템 변수(MM/DD/YYYY) 서식으로 제공된 날짜 필드.
- 트랜잭션을 2개월 세그먼트로 그룹화하고 각 트랜잭션에 대한 해당 세그먼트의 경계 이름을 반환하는 필드 bi_monthly_range 만들기.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

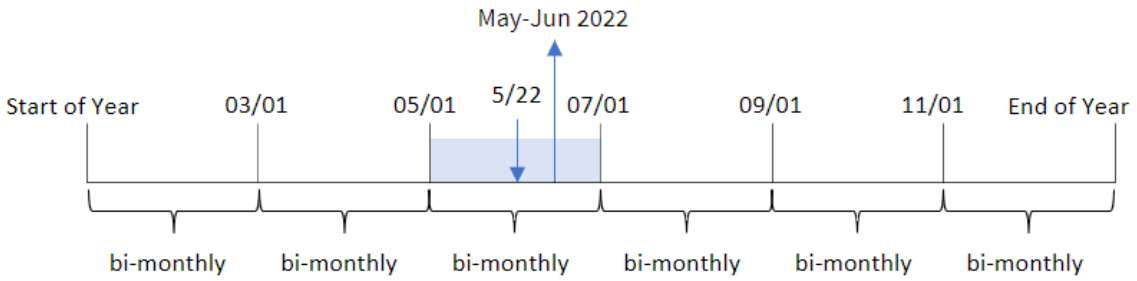
- date
- bi_monthly_range

결과 테이블

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

'bi_monthly_range' 필드는 monthsname() 함수를 사용하여 선행 LOAD 문에서 만들어집니다. 제공된 첫 번째 인수는 2이며, 연도를 2개월 세그먼트로 나눕니다. 두 번째 인수는 평가 중인 필드를 식별합니다.

monthsname 함수의 다이어그램, 기본 예



트랜잭션 8195는 5월 22일에 발생합니다. `monthsname()` 함수는 처음에 연도를 2개월 세그먼트로 나눕니다. 트랜잭션 8195는 5월 1일에서 6월 30일 사이의 세그먼트에 속합니다. 따라서 이 함수는 `MonthNames` 시스템 변수 서식으로 이러한 월과 연도를 반환합니다(May-Jun 2022).

예 2 – period_no

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 첫 번째 예와 동일한 인라인 데이터 집합 및 시나리오.
- 트랜잭션을 2개월 세그먼트로 그룹화하고 각 트랜잭션에 대한 이전 세그먼트 경계 이름을 반환하는 필드 `prev_bi_monthly_range` 만들기.

필요에 따라 목록 등을 사용하여 여기에 다른 텍스트를 추가합니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- `date`
- `prev_bi_monthly_range`

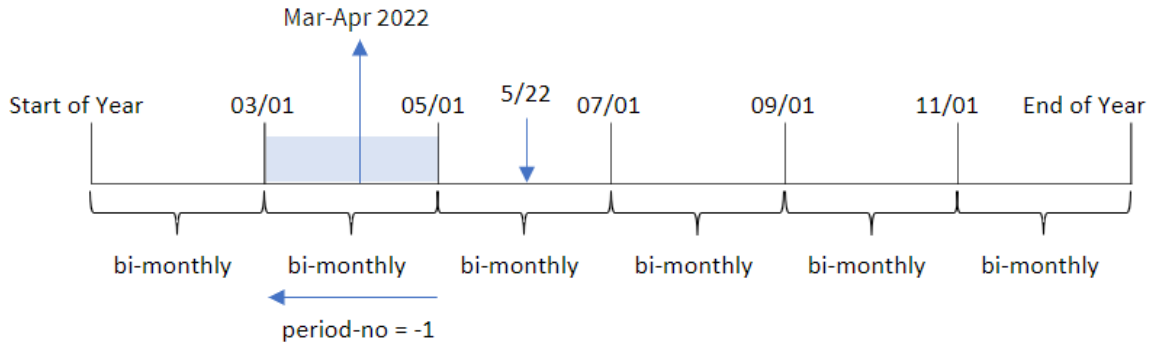
결과 테이블

date	**prev_bi_monthly_range**
2/19/2022	Nov-Dec 2021
3/7/2022	Jan-Feb 2022
3/30/2022	Jan-Feb 2022
4/5/2022	Jan-Feb 2022
4/16/2022	Jan-Feb 2022
5/1/2022	Mar-Apr 2022
5/7/2022	Mar-Apr 2022
5/22/2022	Mar-Apr 2022
6/15/2022	Mar-Apr 2022
6/26/2022	Mar-Apr 2022
7/9/2022	May-Jun 2022
7/22/2022	May-Jun 2022
7/23/2022	May-Jun 2022
7/27/2022	May-Jun 2022
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

이 예에서 -1은 monthsname() 함수의 period_no 인수로 사용됩니다. 처음에 1년을 2개월 세그먼트로 나눈 후 함수는 트랜잭션이 발생할 때 이전 세그먼트 경계를 반환합니다.

monthsname 함수의 다이어그램, period_no 예



트랜잭션 8195는 5월과 6월 사이의 세그먼트에서 발생합니다. 따라서 이전의 2개월 세그먼트는 3월 1일과 4월 30일 사이였으므로 함수는 Mar-Apr 2022를 반환합니다.

예 3 – first_month_of_year

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 첫 번째 예와 동일한 인라인 데이터 집합 및 시나리오.
- 트랜잭션을 2개월 세그먼트로 그룹화하고 각 트랜잭션에 대한 세그먼트 경계 이름을 반환하는 필드 bi_monthly_range 만들기.

그러나 이 예에서는 4월을 회계 연도의 첫 번째 달로 설정해야 합니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- bi_monthly_range

결과 테이블

date	bi_monthly_range
2/19/2022	Feb-Mar 2021
3/7/2022	Feb-Mar 2021
3/30/2022	Feb-Mar 2021
4/5/2022	Apr-May 2022

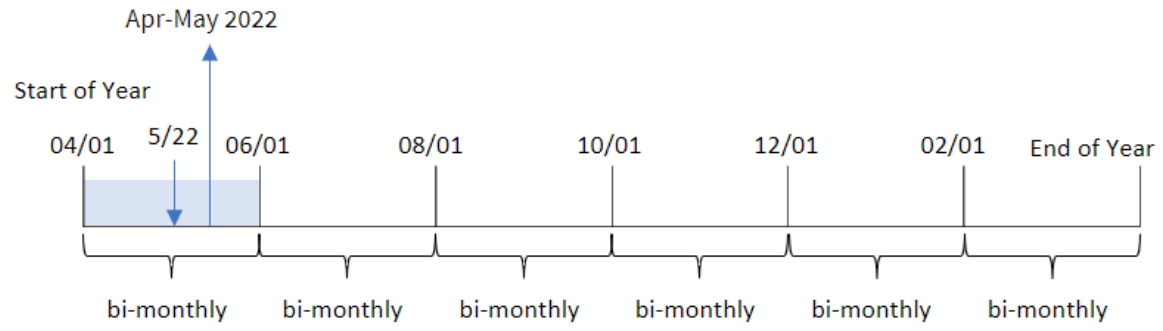
date	bi_monthly_range
4/16/2022	Apr-May 2022
5/1/2022	Apr-May 2022
5/7/2022	Apr-May 2022
5/22/2022	Apr-May 2022
6/15/2022	Jun-Jul 2022
6/26/2022	Jun-Jul 2022
7/9/2022	Jun-Jul 2022
7/22/2022	Jun-Jul 2022
7/23/2022	Jun-Jul 2022
7/27/2022	Jun-Jul 2022
8/2/2022	Aug-Sep 2022
8/8/2022	Aug-Sep 2022
8/19/2022	Aug-Sep 2022
9/26/2022	Aug-Sep 2022
10/14/2022	Oct-Nov 2022
10/29/2022	Oct-Nov 2022

`monthsname()` 함수의 `first_month_of_year` 인수로 4를 사용하면 이 함수는 연도를 4월 1일에 시작합니다. 그런 다음 연도를 2개월 세그먼트로 나눕니다. Apr-May,Jun-Jul,Aug-Sep,Oct-Nov,Dec-Jan,Feb-Mar.

결과에 대한 단락 텍스트입니다.

트랜잭션 8195는 5월 22일에 발생했으며 4월 1일에서 5월 31일 사이에 해당합니다. 따라서 함수는 Apr-May 2022를 반환합니다.

monthsname 함수의 다이어그램, first_month_of_year 예



예 4 – 차트 개체 예

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

이 로드 스크립트에는 첫 번째 예와 동일한 인라인 데이터 집합 및 시나리오가 포함되어 있습니다. 그러나 이 예에서는 변경되지 않은 데이터 집합이 응용 프로그램에 로드됩니다. 트랜잭션을 2개월 세그먼트로 그룹화하고 각 트랜잭션에 대한 세그먼트 경계를 반환하는 계산은 응용 프로그램의 차트 개체에서 측정값으로 만들어집니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다.`date`.

다음 측정값을 만듭니다.

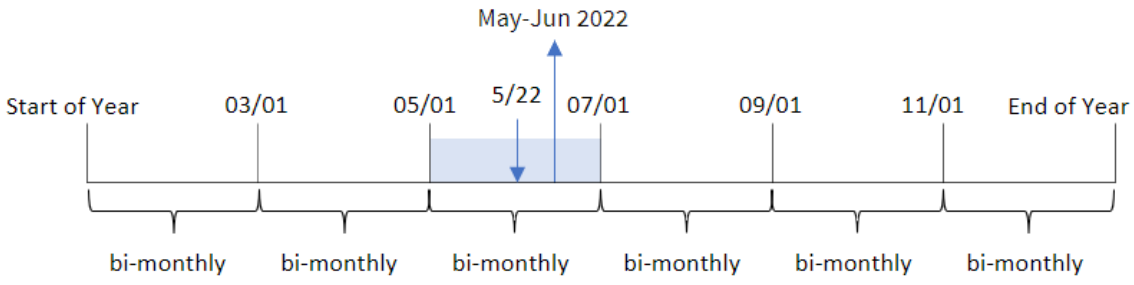
```
=monthsname(2,date)
```

결과 테이블

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

`bi_monthly_range` 필드는 `monthsname()` 함수를 사용하여 차트 개체에서 측정값으로 만들어집니다. 제공된 첫 번째 인수는 2이며, 연도를 2개월 세그먼트로 나눕니다. 두 번째 인수는 평가 중인 필드를 식별합니다.

monthsname 함수의 다이어그램, 차트 개체 예



트랜잭션 8195는 5월 22일에 발생합니다. monthsname() 함수는 처음에 연도를 2개월 세그먼트로 나눕니다. 트랜잭션 8195는 5월 1일에서 6월 30일 사이의 세그먼트에 속합니다. 따라서 이 함수는 MonthNames 시스템 변수 서식으로 이러한 월과 연도를 반환합니다(May-Jun 2022).

예 5 – 시나리오

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Transactions라는 테이블에 로드되는 2022년 트랜잭션이 포함된 데이터 집합.
- DateFormat 시스템 변수(MM/DD/YYYY) 서식으로 제공된 날짜 필드.

최종 사용자는 선택한 기간별로 총 판매액을 표시하는 차트 개체를 원합니다. 이는 변수 입력 컨트롤에 의해 동적으로 수정되는 계산 차원으로 monthsname() 함수를 사용하여 데이터 모델에서 이 차원을 사용할 수 없는 경우에도 구현할 수 있습니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다.

로드 스크립트 시작 시 변수 입력 컨트롤에 연결될 변수(vPeriod)가 만들어졌습니다. 그런 다음 시트에서 변수를 사용자 지정 개체로 구성합니다.

다음과 같이 하십시오.

1. 자산 패널에서 **사용자 지정 개체**를 클릭합니다.
2. **Qlik 대시보드 번들**을 선택하고 **변수 입력** 개체를 만듭니다.
3. 차트 개체의 제목을 입력합니다.
4. **변수**에서 이름으로 **vPeriod**를 선택하고 **드롭다운**으로 표시할 개체를 설정합니다.
5. **값**에서 동적 값을 사용하도록 개체를 구성합니다. 다음을 입력합니다.
   `='1~month|2~bi-month|3~quarter|4~tertial|6~half-year'`

다음으로 결과 테이블을 만듭니다.

다음과 같이 하십시오.

1. 새 테이블을 만들고 다음 계산된 차원을 추가합니다.
   `=monthsname($(vPeriod),date)`
2. 이 측정값을 추가하여 총 판매액을 계산합니다.
   `=sum(amount)`
3. 측정값의 **숫자 형식**을 **화폐**로 설정합니다. ✔ **편집 완료**를 클릭합니다. 이제 변수 개체에서 시간 세그먼트를 조정하여 테이블에 표시된 데이터를 수정할 수 있습니다.

tertial 옵션을 선택하면 결과 테이블이 다음과 같이 표시됩니다.

결과 테이블

monthsname($(vPeriod),date)	=sum(amount)
Jan-Apr 2022	253.89
May-Aug 2022	713.58
Sep-Dec 2022	248.12

monthsstart

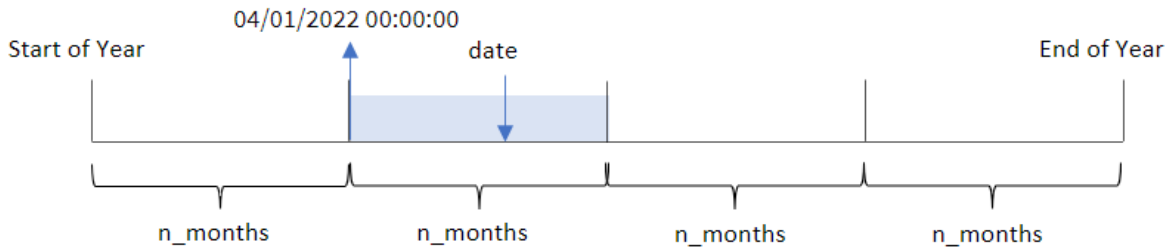
이 함수는 기준일을 포함하는 월, 2개월, 분기, 4개월 기간 또는 6개월의 첫 번째 밀리초의 타임스탬프에 해당하는 값을 반환합니다. 또한 이전 기간 또는 다음 기간에 대한 타임스탬프를 찾을 수도 있습니다.기본 출력 형식은 스크립트에 설정된 **DateFormat**입니다.

구문:

```
MonthsStart(n_months, date[, period_no [, first_month_of_year]])
```

반환 데이터 유형: dual

monthsstart() 함수 다이어그램



monthsstart() 함수는 제공된 n_months 인수를 기반으로 연도를 세그먼트로 나눕니다. 그런 다음 제공된 각 날짜가 속하는 세그먼트를 평가하고 해당 세그먼트의 첫 번째 밀리초를 날짜 서식으로 반환합니다. 이 함수는 또한 이전 또는 다음 세그먼트에서 시작 타임스탬프를 반환하고 연도의 첫 번째 달을 재정의하는 기능을 제공합니다.

함수에서 n_month 인수로 사용할 수 있는 연도의 다음 세그먼트:

가능한 n_month 인수

기간	개월 수
1개월	1
2개월	2
분기	3
4개월	4
6개월	6

인수

인수	설명
n_months	기간을 정의하는 개월 수입니다. 정수 또는 정수로 처리되는 표현식으로, 1(inmonth() 함수와 동일), 2(2개월), 3(inquarter() 함수와 동일), 4(4개월 기간) 또는 6(6개월) 중 하나여야 합니다.
date	평가할 날짜 또는 타임스탬프입니다.
period_no	기간은 **period_no**, 정수 또는 정수로 처리되는 표현식으로 오프셋을 지정할 수 있습니다. 값 0은 **base_date**를 포함하는 기간을 나타냅니다. **period_no**가 음수 값일 경우 이전 기간, 양수 값일 경우 다음 기간을 나타냅니다.
first_month_of_year	1월에 시작되지 않는 (회계)연도를 사용하려는 경우 **first_month_of_year**에 2와 12 사이의 값을 지정하십시오.

사용 시기

monthsstart() 함수는 일반적으로 사용자가 아직 발생하지 않은 기간을 사용하여 계산하려고 할 때 표현식의 일부로 사용됩니다. 예를 들어 사용자가 월, 분기 또는 6개월 동안 누적된 총 이자를 계산할 수 있도록 입력 변수를 제공하는 데 사용할 수 있습니다.

함수 예

예	결과
monthsstart(4, '10/19/2013')	09/01/2013를 반환합니다.
monthsstart(4, '10/19/2013, -1)	05/01/2013을 반환합니다.
monthsstart(4, '10/19/2013', 0, 2)	2개월차에 연도가 시작되므로 10/01/2013을 반환합니다.

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

예 1 – 추가 인수 없음

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Transactions라는 테이블에 로드되는 2022년 트랜잭션 집합이 포함된 데이터 집합.
- DateFormat 시스템 변수(MM/DD/YYYY) 서식으로 제공된 날짜 필드.
- 트랜잭션을 2개월 세그먼트로 그룹화하고 각 트랜잭션에 대한 세그먼트의 시작 타임스탬프를 반환하는 필드 bi_monthly_start 만들기.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- bi_monthly_start
- bi_monthly_start_timestamp

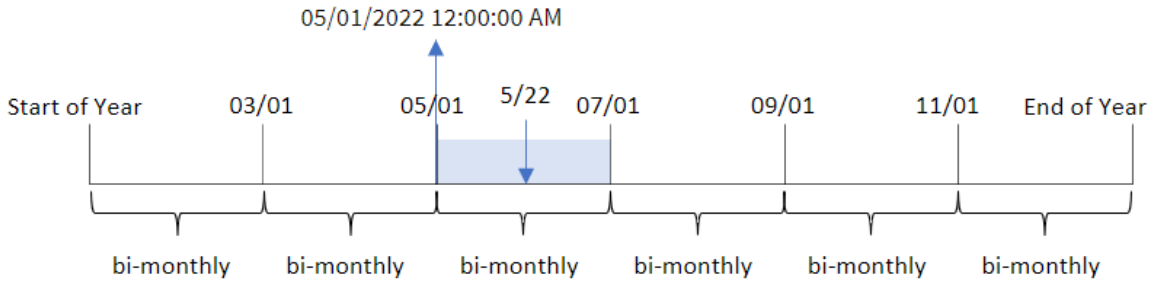
결과 테이블

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

date	bi_monthly_start	bi_monthly_start_timestamp
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

'bi_monthly_start' 필드는 monthsstart() 함수를 사용하여 선행 LOAD 문에서 만들어집니다. 제공된 첫 번째 인수는 2이며, 연도를 2개월 세그먼트로 나눕니다. 두 번째 인수는 평가 중인 필드를 식별합니다.

monthsstart() 함수 다이어그램, 추가 인수가 없는 예



트랜잭션 8195는 5월 22일에 발생합니다. monthsstart() 함수는 처음에 연도를 2개월 세그먼트로 나눕니다. 트랜잭션 8195는 5월 1일에서 6월 30일 사이의 세그먼트에 속합니다. 따라서 이 함수는 이 세그먼트의 첫 번째 밀리초, 즉 2022년 4월 1일 오전 12:00:00를 반환합니다.

예 2 – period_no

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 첫 번째 예와 동일한 데이터 집합 및 시나리오.
- 트랜잭션이 발생하기 전 2개월 세그먼트의 첫 번째 밀리초를 반환하는 필드 prev_bi_monthly_start 만들기.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

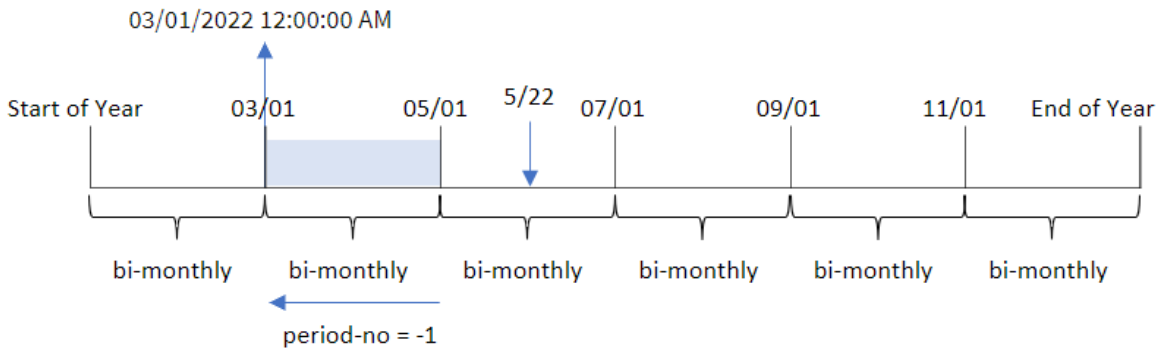
- date
- prev_bi_monthly_start
- prev_bi_monthly_start_timestamp

결과 테이블

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
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

초기에 1년을 2개월 세그먼트로 나눈 후 monthsstart() 함수의 period_no 인수로 -1을 사용하면 이 함수는 트랜잭션이 발생할 때 이전 2개월 세그먼트의 첫 번째 밀리초를 반환합니다.

monthsstart() 함수의 다이어그램, period_no 예



트랜잭션 8195는 5월과 6월 사이의 세그먼트에서 발생합니다. 따라서 이전의 2개월 세그먼트는 3월 1일과 4월 30일 사이에 있었으므로 이 함수는 이 세그먼트의 첫 번째 밀리초인 2022년 3월 1일 오전 12:00:00를 반환합니다.

예 3 – first_month_of_year

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 첫 번째 예와 동일한 데이터 집합 및 시나리오.
- 트랜잭션을 2개월 세그먼트로 그룹화하고 각 트랜잭션에 대한 집합의 시작 타임스탬프를 반환하는 필드 `bi_monthly_start` 만들기.

그러나 이 예에서는 4월을 회계 연도의 첫 번째 달로 설정해야 합니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- bi_monthly_start
- bi_monthly_start_timestamp

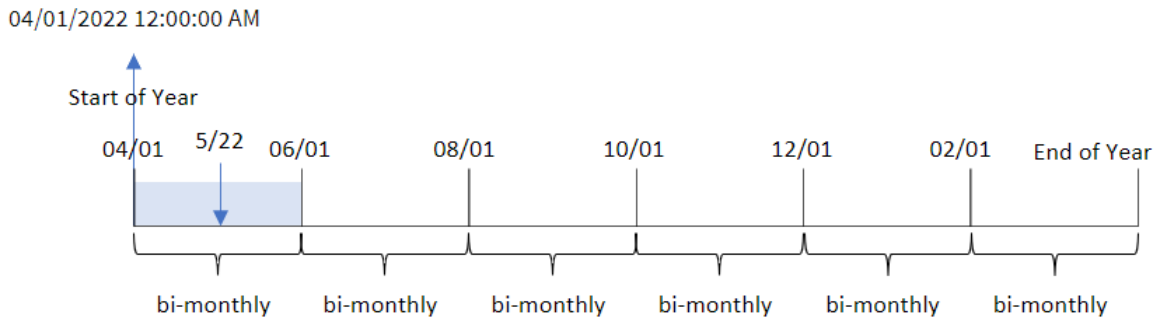
결과 테이블

date	bi_monthly_start	bi_monthly_start_timestamp
2/19/2022	02/01/2022	2/1/2022 12:00:00 AM
3/7/2022	02/01/2022	2/1/2022 12:00:00 AM
3/30/2022	02/01/2022	2/1/2022 12:00:00 AM
4/5/2022	04/01/2022	4/1/2022 12:00:00 AM
4/16/2022	04/01/2022	4/1/2022 12:00:00 AM
5/1/2022	04/01/2022	4/1/2022 12:00:00 AM
5/7/2022	04/01/2022	4/1/2022 12:00:00 AM
5/22/2022	04/01/2022	4/1/2022 12:00:00 AM
6/15/2022	06/01/2022	6/1/2022 12:00:00 AM
6/26/2022	06/01/2022	6/1/2022 12:00:00 AM
7/9/2022	06/01/2022	6/1/2022 12:00:00 AM
7/22/2022	06/01/2022	6/1/2022 12:00:00 AM

date	bi_monthly_start	bi_monthly_start_timestamp
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

`monthsstart()` 함수의 `first_month_of_year` 인수로 4를 사용하면 이 함수는 연도를 4월 1일에 시작합니다. 그런 다음 연도를 2개월 세그먼트로 나눕니다. Apr-May,Jun-Jul,Aug-Sep,Oct-Nov,Dec-Jan,Feb-Mar.

`monthsstart()` 함수의 다이어그램, first_month_of_year 예



트랜잭션 8195는 5월 22일에 발생했으며 4월 1일에서 5월 31일 사이에 해당합니다. 따라서 이 함수는 2022년 4월 1일 오전 12:00:00에 이 세그먼트의 첫 번째 밀리초를 반환합니다.

예 4 – 차트 개체 예

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

이 로드 스크립트에는 첫 번째 예와 동일한 데이터 집합 및 시나리오가 포함되어 있습니다.

그러나 이 예에서는 변경되지 않은 데이터 집합이 응용 프로그램에 로드됩니다. 트랜잭션을 2개월 세그먼트로 그룹화하고 각 트랜잭션에 대한 집합의 시작 타임스탬프를 반환하는 계산은 응용 프로그램의 차트 개체에서 측정값으로 만들어집니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다. date.

다음 측정값을 만듭니다.

```
=monthsstart(2,date)
```

```
=timestamp(monthsstart(2,date))
```

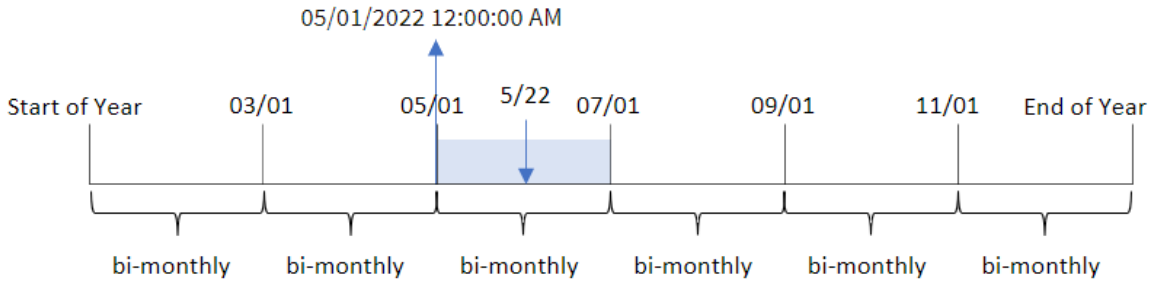
이러한 계산은 각 트랜잭션이 발생한 2개월 세그먼트의 시작 타임스탬프를 검색합니다.

결과 테이블

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

date	=monthsstart(2,date)	=timestamp(monthsstart(2,date))
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

monthsstart() 함수의 다이어그램, 차트 개체 예



트랜잭션 8195는 5월 22일에 발생했습니다. monthsstart() 함수는 처음에 연도를 2개월 세그먼트로 나눕니다. 트랜잭션 8195는 5월 1일에서 6월 30일 사이의 세그먼트에 속합니다. 따라서 이 함수는 이 세그먼트의 첫 번째 밀리초, 즉 2022년 5월 1일 오전 12:00:00를 반환합니다.

예 5 – 시나리오

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Loans라는 테이블에 로드되는 대출 잔액 집합이 포함된 데이터 집합.
- 대출 ID, 월초 잔액, 각 대출에 연간 부과되는 단순 이자율로 구성된 데이터.

최종 사용자는 선택한 기간 동안 각 대출에 대해 발생한 현재 이자를 대출 ID별로 표시하는 차트 개체를 원합니다. 회계 연도는 1월에 시작됩니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다.

로드 스크립트 시작 시 변수 입력 컨트롤에 연결될 변수(vPeriod)가 만들어졌습니다. 그런 다음 시트에서 변수를 사용자 지정 개체로 구성합니다.

다음과 같이 하십시오.

1. 자산 패널에서 **사용자 지정 개체**를 클릭합니다.
2. **Qlik 대시보드 번들**을 선택하고 **변수 입력** 개체를 만듭니다.
3. 차트 개체의 제목을 입력합니다.
4. **변수**에서 이름으로 **vPeriod**를 선택하고 **드롭다운**으로 표시할 개체를 설정합니다.
5. **값**에서 동적 값을 사용하도록 개체를 구성합니다. 다음을 입력합니다.
   ```
   ='1~month|2~bi-month|3~quarter|4~tertial|6~half-year'
   ```

다음으로 결과 테이블을 만듭니다.

다음과 같이 하십시오.

1. 새 테이블을 만듭니다. 다음 필드를 차원으로 추가합니다.
   - employee_id
   - employee_name
2. 누적 이자를 계산하는 측정값을 만듭니다.
   `=start_balance*(rate*(today(1)-monthsstart($(vPeriod),today(1)))/365)`
3. 측정값의 **숫자 형식**을 **화폐**로 설정합니다. ✔ **편집 완료**를 클릭합니다. 이제 변수 개체에서 시간 세그먼트를 조정하여 테이블에 표시된 데이터를 수정할 수 있습니다.

month 기간 옵션을 선택하면 결과 테이블이 다음과 같이 표시됩니다.

결과 테이블

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

사용자의 입력을 첫 번째 인수로 사용하고 오늘 날짜를 두 번째 인수로 사용하는 monthsstart() 함수는 사용자가 선택한 기간의 시작 날짜를 반환합니다. 이 표현식은 현재 날짜에서 해당 결과를 빼서 이 기간에 지금까지 경과한 일수를 반환합니다.

그런 다음 이 값에 이자율을 곱하고 365로 나누어 이 기간 동안 발생한 유효 이자율을 반환합니다. 그런 다음 결과에 대출의 시작 잔액을 곱하여 이 기간의 지금까지 발생한 이자를 반환합니다.

monthstart

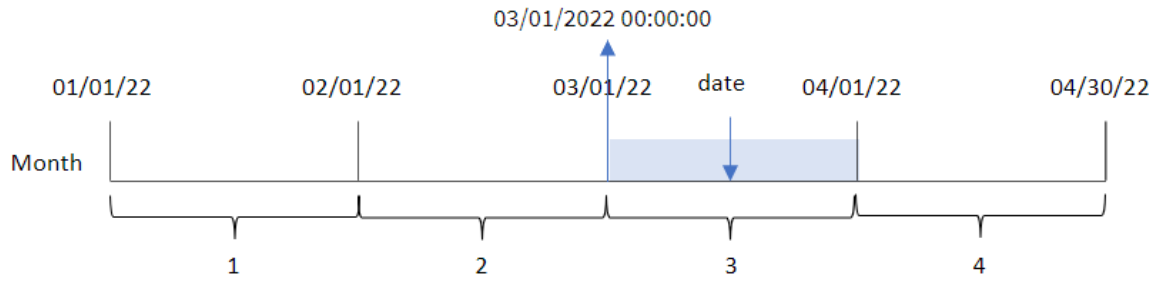
이 함수는 **date**를 포함하는 월의 첫 번째 날의 첫 번째 밀리초의 타임스탬프에 해당하는 값을 반환합니다. 기본 출력 형식은 스크립트에 설정된 **DateFormat**입니다.

구문:

```
MonthStart(date[, period_no])
```

반환 데이터 유형: dual

monthstart() 함수 다이어그램



monthstart() 함수는 날짜가 속하는 월을 확인합니다. 그런 다음 해당 월의 첫 번째 밀리초에 대한 타임스탬프를 날짜 서식으로 반환합니다.

인수

인수	설명
date	평가할 날짜 또는 타임스탬프입니다.
period_no	**period_no**는 정수이며, 0이거나 생략되는 경우는 **date**를 포함하는 월을 나타냅니다. **period_no**가 음수 값일 경우 이전 달, 양수 값일 경우 다음 달을 나타냅니다.

사용 시기

monthstart() 함수는 일반적으로 사용자가 지금까지 경과된 월의 부분을 사용하여 계산하려고 할 때 표현식의 일부로 사용됩니다. 예를 들어 특정 날짜까지 한 달 동안 누적된 이자를 계산하는 데 사용할 수 있습니다.

함수 예

예	결과
monthstart('10/19/2001')	10/01/2001를 반환합니다.
monthstart('10/19/2001', -1)	09/01/2001를 반환합니다.

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

예 1 – 추가 인수 없음

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Transactions라는 테이블에 로드되는 2022년 트랜잭션 집합이 포함된 데이터 집합.
- DateFormat 시스템 변수(MM/DD/YYYY) 서식으로 제공된 날짜 필드.
- 트랜잭션이 발생한 월의 시작에 대한 타임스탬프를 반환하는 필드 start_of_month 만들기.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- start_of_month
- start_of_month_timestamp

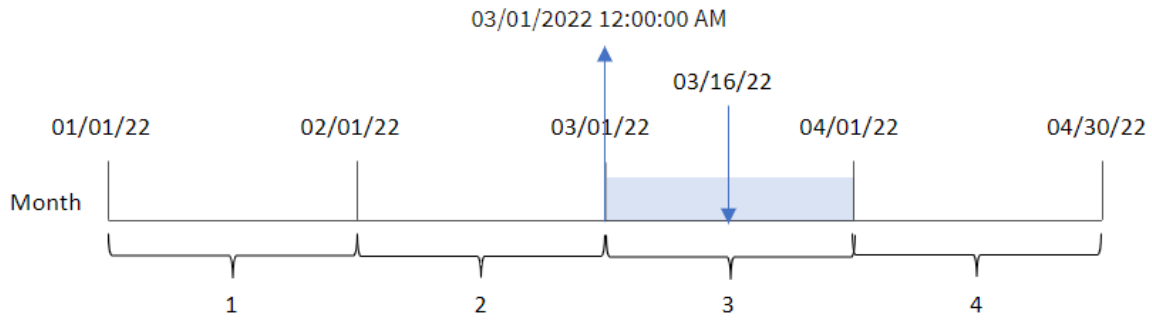
결과 테이블

date	start_of_month	start_of_month_timestamp
1/7/2022	01/01/2022	1/1/2022 12:00:00 AM
1/19/2022	01/01/2022	1/1/2022 12:00:00 AM
2/5/2022	02/01/2022	2/1/2022 12:00:00 AM
2/28/2022	02/01/2022	2/1/2022 12:00:00 AM
3/16/2022	03/01/2022	3/1/2022 12:00:00 AM
4/1/2022	04/01/2022	4/1/2022 12:00:00 AM
5/7/2022	05/01/2022	5/1/2022 12:00:00 AM
5/16/2022	05/01/2022	5/1/2022 12:00:00 AM
6/15/2022	06/01/2022	6/1/2022 12:00:00 AM
6/26/2022	07/01/2022	6/1/2022 12:00:00 AM
7/9/2022	07/01/2022	7/1/2022 12:00:00 AM
7/22/2022	07/01/2022	7/1/2022 12:00:00 AM
7/23/2022	07/01/2022	7/1/2022 12:00:00 AM
7/27/2022	07/01/2022	7/1/2022 12:00:00 AM
8/2/2022	08/01/2022	8/1/2022 12:00:00 AM
8/8/2022	08/01/2022	8/1/2022 12:00:00 AM
8/19/2022	08/01/2022	8/1/2022 12:00:00 AM
9/26/2022	09/01/2022	9/1/2022 12:00:00 AM
10/14/2022	10/01/2022	10/1/2022 12:00:00 AM
10/29/2022	10/01/2022	10/1/2022 12:00:00 AM

start_of_month 필드는 monthstart() 함수를 사용하고 날짜 필드를 함수의 인수로 전달하여 선행 LOAD 문에서 만들어집니다.

monthstart() 함수는 날짜 값이 속하는 월을 식별하고 해당 월의 첫 번째 밀리초에 대한 타임스탬프를 반환합니다.

monthstart() 함수 다이어그램, 추가 인수가 없는 예



트랜잭션 8192는 3월 16일에 발생했습니다. monthstart() 함수는 해당 월의 첫 번째 밀리초, 즉 3월 1일 오전 12:00:00를 반환합니다.

예 2 – period_no

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 첫 번째 예와 동일한 데이터 집합 및 시나리오.
- 트랜잭션이 발생하기 전 월의 시작에 대한 타임스탬프를 반환하는 필드 previous_month_start 만들기.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- previous_month_start
- previous_month_start_timestamp

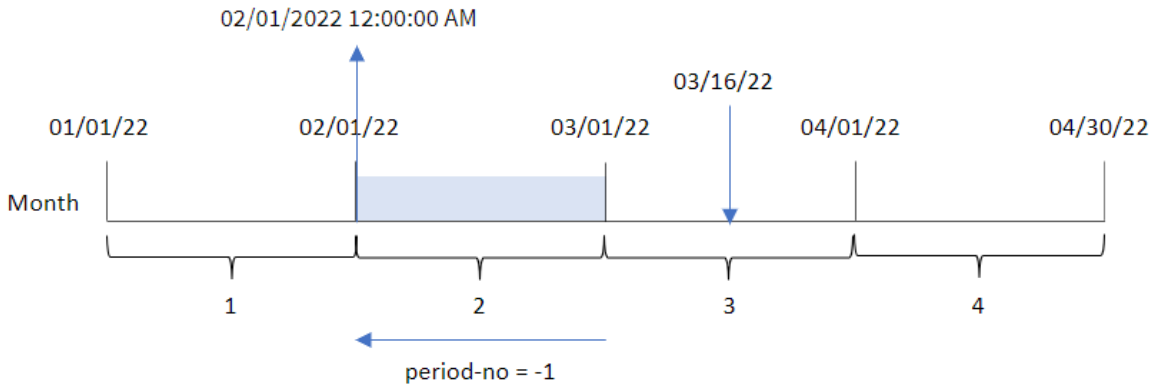
결과 테이블

date	previous_month_start	previous_month_start_timestamp
1/7/2022	12/01/2021	12/1/2021 12:00:00 AM
1/19/2022	12/01/2021	12/1/2021 12:00:00 AM
2/5/2022	01/01/2022	1/1/2022 12:00:00 AM
2/28/2022	01/01/2022	1/1/2022 12:00:00 AM
3/16/2022	02/01/2022	2/1/2022 12:00:00 AM
4/1/2022	03/01/2022	3/1/2022 12:00:00 AM
5/7/2022	04/01/2022	4/1/2022 12:00:00 AM
5/16/2022	04/01/2022	4/1/2022 12:00:00 AM
6/15/2022	05/01/2022	5/1/2022 12:00:00 AM
6/26/2022	05/01/2022	5/1/2022 12:00:00 AM
7/9/2022	06/01/2022	6/1/2022 12:00:00 AM
7/22/2022	06/01/2022	6/1/2022 12:00:00 AM
7/23/2022	06/01/2022	6/1/2022 12:00:00 AM
7/27/2022	06/01/2022	6/1/2022 12:00:00 AM
8/2/2022	07/01/2022	7/1/2022 12:00:00 AM

date	previous_month_start	previous_month_start_timestamp
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

이 경우 monthstart() 함수에서 period_no -1을 오프셋 인수로 사용했으므로 이 함수는 먼저 트랜잭션이 발생한 월을 식별합니다. 그런 다음 한 달 전으로 시프트하고 해당 월의 첫 번째 밀리초를 식별합니다.

monthstart() 함수의 다이어그램, period_no 예



트랜잭션 8192는 3월 16일에 발생했습니다. monthstart() 함수는 트랜잭션이 발생하기 전 월이 2월임을 식별합니다. 그런 다음 해당 월의 첫 번째 밀리초, 즉 2월 1일 오전 12:00:00를 반환합니다.

예 3 – 차트 개체 예

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

이 로드 스크립트에는 첫 번째 예와 동일한 데이터 집합 및 시나리오가 포함되어 있습니다.

그러나 이 예에서는 변경되지 않은 데이터 집합이 응용 프로그램에 로드됩니다. 트랜잭션이 발생한 월의 시작에 대한 타임스탬프를 반환하는 계산은 응용 프로그램의 차트 개체에서 측정값으로 만들어집니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다. `date`.

트랜잭션이 발생한 월의 시작 날짜를 계산하려면 다음 측정값을 만듭니다.

- `=monthstart(date)`
- `=timestamp(monthstart(date))`

결과 테이블

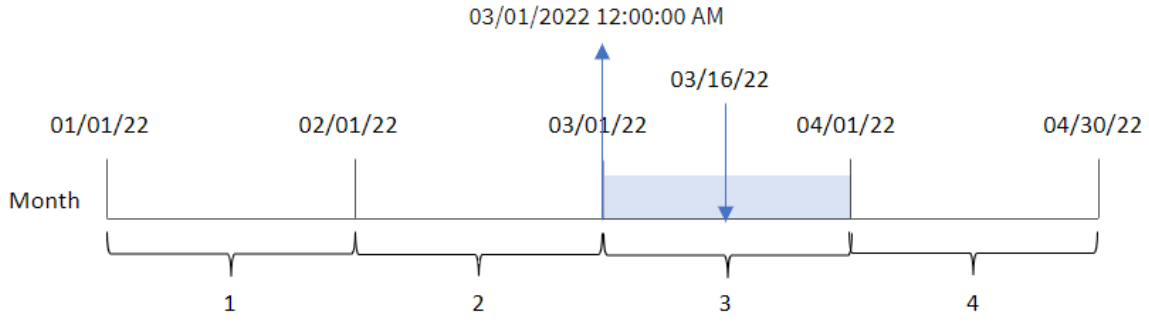
date	=monthstart(date)	=timestamp(monthstart(date))
10/14/2022	10/01/2022	10/1/2022 12:00:00 AM
10/29/2022	10/01/2022	10/1/2022 12:00:00 AM
9/26/2022	09/01/2022	9/1/2022 12:00:00 AM
8/2/2022	08/01/2022	8/1/2022 12:00:00 AM
8/8/2022	08/01/2022	8/1/2022 12:00:00 AM
8/19/2022	08/01/2022	8/1/2022 12:00:00 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

date	=monthstart(date)	=timestamp(monthstart(date))
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

start_of_month 측정값은 monthstart() 함수를 사용하고 날짜 필드를 함수의 인수로 전달하여 차트 개체에서 만들어집니다.

monthstart() 함수는 날짜 값이 속하는 월을 식별하고 해당 월의 첫 번째 밀리초에 대한 타임스탬프를 반환합니다.

monthstart() 함수의 다이어그램, 차트 개체 예



트랜잭션 8192는 3월 16일에 발생했습니다. monthstart() 함수는 트랜잭션이 3월에 발생했음을 식별하고 해당 월의 첫 번째 밀리초, 즉 3월 1일 오전 12:00:00를 반환합니다.

예 4 – 시나리오

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Loans라는 테이블에 로드되는 대출 잔액 집합이 포함된 데이터 집합.
- 대출 ID, 월초 잔액, 각 대출에 연간 부과되는 단순 이자율로 구성된 데이터.

최종 사용자는 해당 월부터 현재까지 각 대출에 대해 발생한 현재 이자를 대출 ID별로 표시하는 차트 개체를 원합니다.

로드 스크립트

```
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
];
```

결과

다음과 같이 하십시오.

1. 데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.
   - loan_id
   - start_balance
2. 다음으로 누적 이자를 계산하는 측정값을 만듭니다.
   `=start_balance*(rate*(today(1)-monthstart(today(1)))/365)`
3. 측정값의 **숫자 형식**을 **화폐**로 설정합니다.

결과 테이블

loan_id	start_balance	=start_balance*(rate*(today(1)-monthstart(today(1)))/365)
8188	$10000.00	$16.44

loan_id	start_balance	=start_balance*(rate*(today(1)-monthstart(today(1)))/365)
8189	$15000.00	$58.56
8190	$17500.00	$28.77
8191	$21000.00	$48.90
8192	$90000.00	$517.81

오늘 날짜를 유일한 인수로 사용하는 monthstart() 함수는 현재 월의 시작 날짜를 반환합니다. 이 표현식은 현재 날짜에서 해당 결과를 빼서 이번 달에 지금까지 경과한 일수를 반환합니다.

그런 다음 이 값에 이자율을 곱하고 365로 나누어 이 기간 동안 발생한 유효 이자율을 반환합니다. 그런 다음 결과에 대출의 시작 잔액을 곱하여 이번 달에 지금까지 발생한 이자를 반환합니다.

networkdays

networkdays 함수는 선택적으로 나열된 **holiday**를 고려하여 **start_date**(포함)와 **end_date**(포함) 사이의 근무일(월요일 ~ 금요일)의 수를 반환합니다.

구문:

```
networkdays (start_date, end_date [, holiday])
```

반환 데이터 유형: 정수

networkdays 함수에서 반환된 날짜 범위를 표시하는 캘린더 다이어그램

networkdays 함수에는 다음과 같은 제한 사항이 있습니다.

- 근무일을 수정하는 방법이 없습니다. 즉, 월요일부터 금요일까지 근무하는 것 외에는 지역이나 상황에 따라 함수를 수정할 수 있는 방법이 없습니다.
- holiday 매개 변수는 문자열 상수여야 합니다. 표현식은 허용되지 않습니다.

인수

인수	설명
start_date	평가할 시작 날짜입니다.
end_date	평가할 끝 날짜입니다.
holiday	근무일에서 제외시킬 공휴일 기간입니다. 휴일은 문자열 상수 날짜로 표시됩니다. 여러 휴일 날짜를 쉼표로 구분하여 지정할 수 있습니다. **'12/25/2013', '12/26/2013', '12/31/2013', '01/01/2014'**

사용 시기

networkdays() 함수는 일반적으로 사용자가 두 날짜 사이에 발생하는 근무 일수를 사용하여 계산하려고 할 때 표현식의 일부로 사용됩니다. 예를 들어, 사용자가 PAYE(Pay-as-you-earn) 계약에 따라 직원이 받을 총 임금을 계산하려는 경우입니다.

함수 예

예	결과
networkdays ('12/19/2013', '01/07/2014')	14을 반환합니다. 이 예에서는 공휴일을 고려하지 않습니다.
networkdays ('12/19/2013', '01/07/2014', '12/25/2013', '12/26/2013')	12을 반환합니다. 이 예에서는 12/25/2013 ~ 12/26/2013의 공휴일을 고려합니다.
networkdays ('12/19/2013', '01/07/2014', '12/25/2013', '12/26/2013', '12/31/2013', '01/01/2014')	10을 반환합니다. 이 예에서는 2번의 공휴일 기간을 고려합니다.

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

예 1 – 기본 예

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 프로젝트 ID, 시작 날짜 및 끝 날짜가 포함된 데이터 집합. 이 정보는 Projects라는 테이블에 로드됩니다.
- DateFormat 시스템 변수(MM/DD/YYYY) 서식으로 제공된 날짜 필드.
- 각 프로젝트에 관련된 근무일 수를 계산하기 위해 추가 필드 net_work_days 만들기.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- id
- start_date
- end_date
- net_work_days

결과 테이블

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

예정된 휴일(networkdays() 함수의 세 번째 인수에 있었어야 함)이 없으므로, 이 함수는 end_date에서 start_date를 빼고 모든 주말 수도 제외하여 두 날짜 사이의 근무일 수를 계산합니다.

프로젝트 5의 근무일을 강조하는 캘린더 다이어그램(휴일 없음)

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			

위의 캘린더는 id가 5인 프로젝트를 시각적으로 설명합니다. 프로젝트 5는 2022년 8월 10일 수요일에 시작하여 2022년 8월 26일에 끝납니다. 모든 토요일과 일요일을 무시하면 이 두 날짜 사이에 13일의 근무일이 있습니다.

예 2 – 단일 휴일

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 이전 예와 동일한 데이터 집합 및 시나리오.
- DateFormat 시스템 변수(MM/DD/YYYY) 서식으로 제공된 날짜 필드.
- 각 프로젝트에 관련된 근무일 수를 계산하기 위해 추가 필드 net_work_days 만들기.

이 예에서는 2022년 8월 19일에 1일 휴일이 예정되어 있습니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- id
- start_date
- end_date
- net_work_days

결과 테이블

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

networkdays() 함수의 세 번째 인수로 예정된 단일 휴일이 입력됩니다.

프로젝트 5의 근무일을 강조 표시하는 캘린더 다이어그램(단일 휴일)

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			

위의 캘린더에는 프로젝트 5를 시각적으로 요약하여 이 휴일을 포함하도록 조정되어 있습니다. 프로젝트 5에서는 2022년 8월 19일 금요일에 이 휴일이 발생합니다. 결과적으로 프로젝트 5의 총 net_work_days 값은 13일에서 12일로 하루 감소합니다.

예 3 – 여러 휴일

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 첫 번째 예와 동일한 데이터 집합 및 시나리오.
- DateFormat 시스템 변수(MM/DD/YYYY) 서식으로 제공된 날짜 필드.
- 각 프로젝트에 관련된 근무일 수를 계산하기 위해 추가 필드 net_work_days 만들기.

그러나 이 예에서는 2022년 8월 18일부터 8월 21일까지 4개의 휴일이 예정되어 있습니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- id
- start_date
- end_date
- net_work_days

결과 테이블

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

4개의 예정된 휴일이 networkdays() 함수의 세 번째 인수에서 쉼표로 구분된 목록으로 입력됩니다.

프로젝트 5의 근무일을 강조 표시한 캘린더 다이어그램(여러 휴일)

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			

위의 캘린더에는 프로젝트 5를 시각적으로 요약하여 이러한 휴일을 포함하도록 조정되어 있습니다. 프로젝트 5에서 이 예정된 휴일 기간이 발생하며, 그중 2일은 목요일과 금요일에 발생합니다. 결과적으로 프로젝트 5의 총 net_work_days 값은 13일에서 11일로 감소합니다.

예 4 – 단일 휴일

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 첫 번째 예와 동일한 데이터 집합 및 시나리오.
- DateFormat 시스템 변수(MM/DD/YYYY) 서식으로 제공된 날짜 필드.

2022년 8월 19일에 1일 휴일이 예정되어 있습니다.

그러나 이 예에서는 변경되지 않은 데이터 집합이 응용 프로그램에 로드됩니다. net_work_days 필드는 차트 개체의 측정값으로 계산됩니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- id
- start_date
- end_date

다음 측정값을 만듭니다.

```
= networkdays(start_date,end_date,'08/19/2022')
```

결과 테이블

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

networkdays() 함수의 세 번째 인수로 예정된 단일 휴일이 입력됩니다.

단일 휴일이 있는 순 근무일을 보여 주는 캘린더 다이어그램(차트 개체)

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			

위의 캘린더에는 프로젝트 5를 시각적으로 요약하여 이 휴일을 포함하도록 조정되어 있습니다. 프로젝트 5에서는 2022년 8월 19일 금요일에 이 휴일이 발생합니다. 결과적으로 프로젝트 5의 총 net_work_days 값은 13일에서 12일로 하루 감소합니다.

now

이 함수는 현재 시간의 타임스탬프를 반환합니다. 이 함수는 **TimeStamp** 시스템 변수 서식으로 값을 반환합니다. 기본 **timer_mode** 값은 1입니다.

구문:

```
now([ timer_mode])
```

반환 데이터 유형: dual

now() 함수는 로드 스크립트 또는 차트 개체에서 사용할 수 있습니다.

인수

인수	설명
timer_mode	다음과 같은 값을 가질 수 있습니다. 0(데이터 로드가 마지막으로 완료된 시간) 1(함수 호출 시간) 2(앱이 열린 시간) *데이터 로드 스크립트에서 이 함수를 사용하면 **timer_mode=0**이 마지막으로 데이터 로드가 완료된 시간이 되지만 **timer_mode=1**은 현재 데이터 로드 시 함수 호출 시간을 제공합니다.*

now() 함수는 성능에 영향을 미치므로 함수가 테이블의 표현식 내에서 사용되는 경우 스크롤 문제가 발생할 수 있습니다. 사용이 반드시 필요한 경우가 아니면 today() 함수를 대신 사용하는 것이 좋습니다. 레이아웃에서 now()의 사용이 필요한 경우 가능하면 기본 설정이 아닌 now(0) 또는 now(2)을 사용하는 것이 좋습니다. 지속적인 재계산이 필요하지 않습니다.

사용 시기

now() 함수는 일반적으로 표현식 내에서 구성 요소로 사용됩니다. 예를 들어 제품 수명 주기의 남은 시간을 계산하는 데 사용할 수 있습니다. 표현식에 하루의 일부를 사용해야 하는 경우 today() 함수 대신 now() 함수가 사용됩니다.

다음 표는 timer_mode 인수에 대해 다른 값이 주어졌을 때 now() 함수에서 반환된 결과에 대한 설명을 제공합니다.

함수 예

timer_mode 값	로드 스크립트에서 사용된 경우 결과	차트 개체에서 사용된 경우 결과
0	최신 데이터 다시 로드 이전에 마지막으로 성공한 데이터 다시 로드의 타임스탬프를 TimeStamp 시스템 변수 형식으로 반환합니다.	최신 데이터를 다시 로드한 타임스탬프를 TimeStamp 시스템 변수 형식으로 반환합니다.
1	최신 데이터를 다시 로드한 타임스탬프를 TimeStamp 시스템 변수 형식으로 반환합니다.	함수 호출의 타임스탬프를 TimeStamp 시스템 변수 형식으로 반환합니다.
2	응용 프로그램에서 사용자의 세션이 시작된 타임스탬프를 TimeStamp 시스템 변수 형식으로 반환합니다. 사용자가 스크립트를 다시 로드하지 않으면 업데이트되지 않습니다.	응용 프로그램에서 사용자의 세션이 시작된 타임스탬프를 TimeStamp 시스템 변수 형식으로 반환합니다. 이는 새 세션이 시작되거나 응용 프로그램의 데이터가 다시 로드되면 새로 고쳐집니다.

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

예 1 – 로드 스크립트를 사용한 개체 생성

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

이 예에서는 now() 함수를 사용하여 세 개의 변수를 만듭니다. 각 변수는 timer_mode 옵션 중 하나를 사용하여 효과를 보여 줍니다.

변수가 목적을 나타내려면 스크립트를 다시 로드한 다음 짧은 기간 후에 스크립트를 두 번째로 다시 로드합니다. 그러면 now(0) 및 now(1) 변수가 다른 값을 표시하여 목적을 올바르게 보여 줍니다.

로드 스크립트

```
LET vPreviousDataLoad = now(0);
LET vCurrentDataLoad = now(1);
LET vApplicationOpened = now(2);
```

결과

데이터가 두 번째로 로드되면 아래 지침을 사용하여 3개의 텍스트 상자를 만듭니다.

먼저 이전에 로드된 데이터에 대한 텍스트 상자를 만듭니다.

다음과 같이 하십시오.

1. **텍스트 및 이미지** 차트 개체를 사용하여 텍스트 상자를 만듭니다.
2. 개체에 다음 측정값을 추가합니다.
   =vPreviousDataLoad
3. **모양** 아래에서 **Show titles**를 선택하고 '이전 다시 로드 시간'이라는 제목을 개체에 추가합니다.

다음으로 현재 로드 중인 데이터에 대한 텍스트 상자를 만듭니다.

다음과 같이 하십시오.

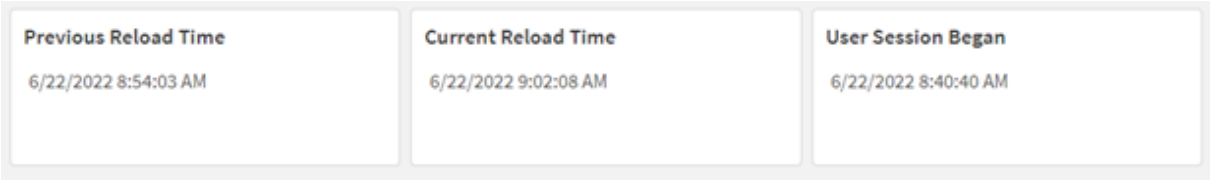
1. **텍스트 및 이미지** 차트 개체를 사용하여 텍스트 상자를 만듭니다.
2. 개체에 다음 측정값을 추가합니다.
   `=vCurrentDataLoad`
3. **모양** 아래에서 **Show titles**를 선택하고 '현재 다시 로드 시간'이라는 제목을 개체에 추가합니다.

응용 프로그램에서 사용자 세션이 시작된 시간을 표시할 최종 텍스트 상자를 만듭니다.

다음과 같이 하십시오.

1. **텍스트 및 이미지** 차트 개체를 사용하여 텍스트 상자를 만듭니다.
2. 개체에 다음 측정값을 추가합니다.
   `=vApplicationOpened`
3. **모양** 아래에서 **Show titles**를 선택하고 '사용자 세션 시작됨'이라는 제목을 개체에 추가합니다.

now() 로드 스크립트 변수



위 이미지는 만들어진 각 변수 값에 대한 예를 보여 줍니다. 예를 들어 값은 다음과 같을 수 있습니다.

- 이전 다시 로드 시간: 6/22/2022 8:54:03 AM
- 현재 다시 로드 시간: 6/22/2022 9:02:08 AM
- 사용자 세션 시작: 6/22/2022 8:40:40 AM

예 2 – 로드 스크립트 없이 개체 생성

로드 스크립트 및 차트 표현식

개요

이 예에서는 변수나 데이터를 응용 프로그램에 로드하지 않고 `now()` 함수를 사용하여 세 개의 차트 개체를 만듭니다. 각 차트 개체는 `timer_mode` 옵션 중 하나를 사용하여 효과를 보여 줍니다.

이 예에는 로드 스크립트가 없습니다.

다음과 같이 하십시오.

1. 데이터 로드 편집기를 엽니다.
2. 기존 로드 스크립트를 변경하지 않고 **데이터 로드**를 클릭합니다.
3. 잠시 후 스크립트를 두 번째로 로드합니다.

결과

데이터가 두 번째로 로드되면 세 개의 텍스트 상자를 만듭니다.

먼저 최신 데이터 다시 로드를 위한 텍스트 상자를 만듭니다.

다음과 같이 하십시오.

1. **텍스트 및 이미지** 차트 개체를 사용하여 텍스트 상자를 만듭니다.
2. 다음 측정값을 추가합니다.
   `=now(0)`
3. **모양**에서 **제목 표시**를 선택하고 개체에 '최신 데이터 다시 로드'라는 제목을 추가합니다.

다음으로 현재 시간을 표시하는 텍스트 상자를 만듭니다.

다음과 같이 하십시오.

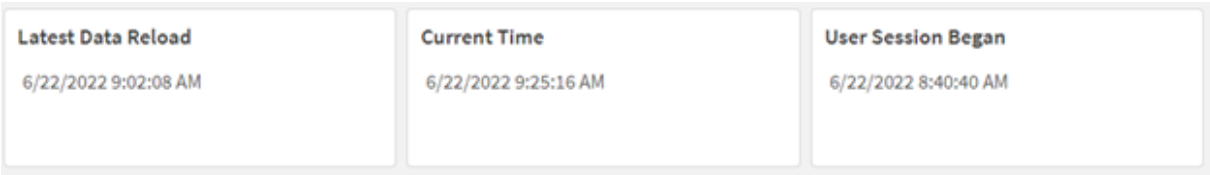
1. **텍스트 및 이미지** 차트 개체를 사용하여 텍스트 상자를 만듭니다.
2. 다음 측정값을 추가합니다.
   `=now(1)`
3. **모양**에서 **제목 표시**를 선택하고 개체에 '현재 시간'이라는 제목을 추가합니다.

응용 프로그램에서 사용자 세션이 시작된 시간을 표시할 최종 텍스트 상자를 만듭니다.

다음과 같이 하십시오.

1. **텍스트 및 이미지** 차트 개체를 사용하여 텍스트 상자를 만듭니다.
2. 다음 측정값을 추가합니다.
   `=now(2)`
3. **모양**에서 **제목 표시**를 선택하고 개체에 '사용자 세션 시작'이라는 제목을 추가합니다.

now() 차트 개체 예



위 이미지는 만들어진 각 개체에 대한 예시 값을 보여 줍니다. 예를 들어 값은 다음과 같을 수 있습니다.

- 최신 데이터 다시 로드: 6/22/2022 9:02:08 AM
- 현재 시간: 6/22/2022 9:25:16 AM
- 사용자 세션 시작: 6/22/2022 8:40:40 AM

'최신 데이터 다시 로드' 차트 개체는 `timer_mode` 값 0을 사용합니다. 이는 데이터가 성공적으로 다시 로드된 마지막 시간의 타임스탬프를 반환합니다.

'현재 시간' 차트 개체는 timer_mode 값 1을 사용합니다. 이는 시스템 시계에 따라 현재 시간을 반환합니다. 시트 또는 개체를 새로 고치면 이 값이 업데이트됩니다.

'사용자 세션 시작' 차트 개체는 timer_mode 값 2를 사용합니다. 이는 응용 프로그램이 열리고 사용자의 세션이 시작된 시간에 대한 타임스탬프를 반환합니다.

예 3 – 시나리오

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Inventory라는 테이블에 로드되는, 암호화폐 채굴 작업에 대한 인벤토리로 구성된 데이터 집합.
- 다음 필드가 있는 데이터: id, purchase_date 및 wph(시간당 와트).

사용자는 해당 월의 지금까지 각 채굴 장비에서 발생한 총 비용을 전력 소비 측면에서 id별로 표시하는 테이블을 원합니다.

이 값은 차트 개체를 새로 고칠 때마다 업데이트되어야 합니다. 현재 전기 요금은 kWH당 $0.0678입니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다. id.

다음 측정값을 만듭니다.

```
=(now(1)-monthstart(now(1)))*24*wph/1000*0.0678
```

차트 개체가 2022년 6월 22일 오전 10:39:05에 새로 고쳐지면 다음 결과가 반환됩니다.

결과 테이블

id	=(now(1)-monthstart(now(1)))*24*wph/1000*0.0678
8188	$39.18
8189	$49.97
8190	$42.81
8191	$46.13
8192	$44.42
8193	$39.18
8194	$46.83
8195	$81.72
8196	$42.95
8197	$42.95
8198	$39.18
8199	$42.29
8200	$42.67
8201	$42.99
8202	$42.99
8203	$42.25
8204	$43.37
8205	$46.13
8206	$39.53

사용자는 개체를 새로 고칠 때마다 개체 결과가 새로 고쳐지기를 원합니다. 따라서 표현식의 now() 함수 인스턴스에 timer_mode 인수가 제공됩니다. now() 함수를 사용하여 monthstart() 함수의 타임스탬프 인수로 식별되는 월 시작에 대한 타임스탬프를 now() 함수로 식별되는 현재 시간에서 뺍니다. 이번 달에 지금까지 경과한 총 시간(일)을 제공합니다.

이 값에 24(하루의 시간)를 곱한 다음 wph 필드의 값을 곱합니다.

시간당 와트에서 시간당 킬로와트로 변환하려면 결과를 1000으로 나눈 다음 제공된 kWH 비율을 곱하면 됩니다.

quarterend

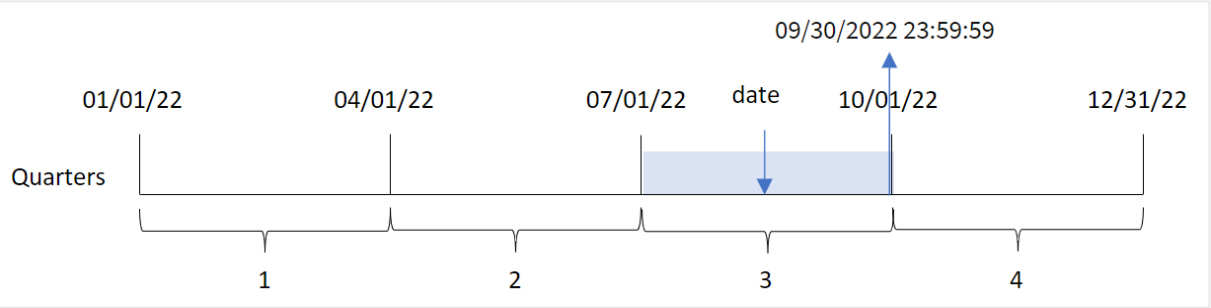
이 함수는 **date**를 포함하는 분기의 마지막 밀리초의 타임스탬프에 해당하는 값을 반환합니다. 기본 출력 형식은 스크립트에 설정된 **DateFormat**입니다.

구문:

```
QuarterEnd(date[, period_no[, first_month_of_year]])
```

반환 데이터 유형: dual

quarterend() 함수 다이어그램



quarterend() 함수는 날짜가 속하는 분기를 확인합니다. 그런 다음 해당 분기의 마지막 달의 마지막 밀리초에 대한 타임스탬프를 날짜 서식으로 반환합니다. 연도의 첫 번째 달은 기본적으로 1월입니다. 그러나 quarterend() 함수의 first_month_of_year 인수를 사용하여 첫 번째로 설정되는 월을 변경할 수 있습니다.

> *quarterend() 함수는 FirstMonthOfYear 시스템 변수를 고려하지 않습니다. first_month_of_year 인수를 사용하여 변경하지 않는 한 연도는 1월 1일에 시작됩니다.*

사용 시기

quarterend() 함수는 일반적으로 아직 발생하지 않은 분기의 부분을 사용하도록 계산하려는 경우 표현식의 일부로 사용됩니다. 예를 들어 해당 분기에 아직 발생하지 않은 총 이자를 계산하려는 경우 사용할 수 있습니다.

인수

인수	설명
date	평가할 날짜 또는 타임스탬프입니다.
period_no	**period_no**는 정수이며, 값 0은 **date**를 포함하는 분기를 나타냅니다. **period_no**가 음수 값일 경우 이전 분기, 양수 값일 경우 다음 분기를 나타냅니다.

인수	설명
first_month_of_year	1월에 시작되지 않는 (회계)연도를 사용하려는 경우 **first_month_of_year**에 2와 12 사이의 값을 지정하십시오.

다음 값을 사용하여 first_month_of_year 인수에서 연도의 첫 번째 달을 설정할 수 있습니다.

first_month_of_year 값

Month	Value
2월	2
3월	3
4월	4
5월	5
6월	6
7월	7
8월	8
9월	9
10월	10
11월	11
12월	12

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

함수 예

예	결과
`quarterend('10/29/2005')`	12/31/2005 23:59:59를 반환합니다.
`quarterend('10/29/2005', -1)`	09/30/2005 23:59:59를 반환합니다.
`quarterend('10/29/2005', 0, 3)`	11/30/2005 23:59:59를 반환합니다.

예 1 - 기본 예

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 'Transactions'라는 테이블에 로드되는 2022년의 트랜잭션 집합을 포함하는 데이터 집합.
- 다음을 포함하는 선행 LOAD:
    - 'end_of_quarter' 필드로 설정되고 트랜잭션이 발생한 분기 끝의 타임스탬프를 반환하는 quarterend() 함수.
    - 'end_of_quarter_timestamp' 필드로 설정되어 선택한 분기의 끝 부분에 대한 정확한 타임스탬프를 반환하는 timestamp() 함수.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- id
- date
- end_of_quarter
- end_of_quarter_timestamp

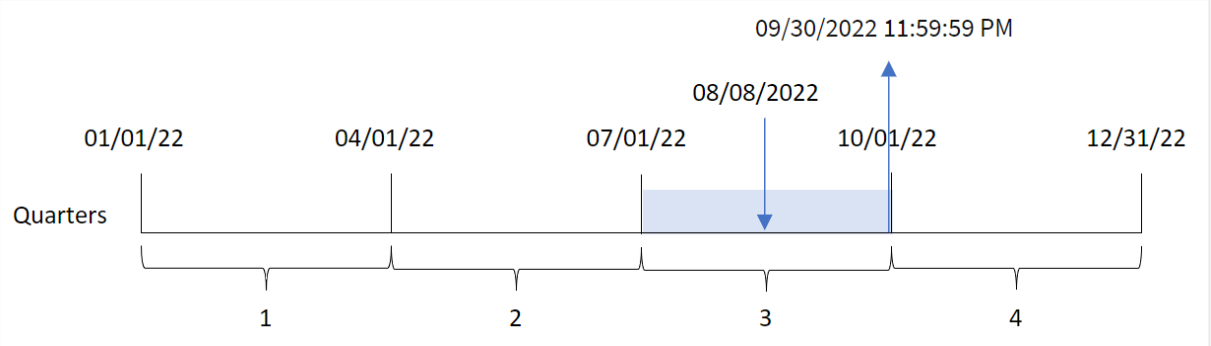
결과 테이블

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

'end_of_quarter' 필드는 quarterend() 함수를 사용하고 날짜 필드를 함수의 인수로 전달하여 선행 LOAD 문에서 만들어집니다.

quarterend() 함수는 초기에 날짜 값이 속하는 분기를 식별한 다음 해당 분기의 마지막 밀리초에 대한 타임스탬프를 반환합니다.

트랜잭션 8203의 분기 끝이 식별된 quarterend() 함수의 다이어그램



트랜잭션 8203은 8월 8일에 발생했습니다. quarterend() 함수는 트랜잭션이 3분기에 발생했음을 식별하고 해당 분기의 마지막 밀리초인 9월 30일 오후 11:59:59를 반환합니다.

예 2 – period_no

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 'Transactions'라는 테이블에 로드되는 2022년의 트랜잭션 집합을 포함하는 데이터 집합.
- 다음을 포함하는 선행 LOAD:
  - 'previous_quarter_end' 필드로 설정되고 트랜잭션이 발생하기 전 분기 끝의 타임스탬프를 반환하는 quarterend()함수.
  - 'previous_end_of_quarter_timestamp' 필드로 설정되고 트랜잭션이 발생하기 전 분기 끝의 정확한 타임스탬프를 반환하는 timestamp()함수.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- id
- date
- previous_quarter_end
- previous_quarter_end_timestamp

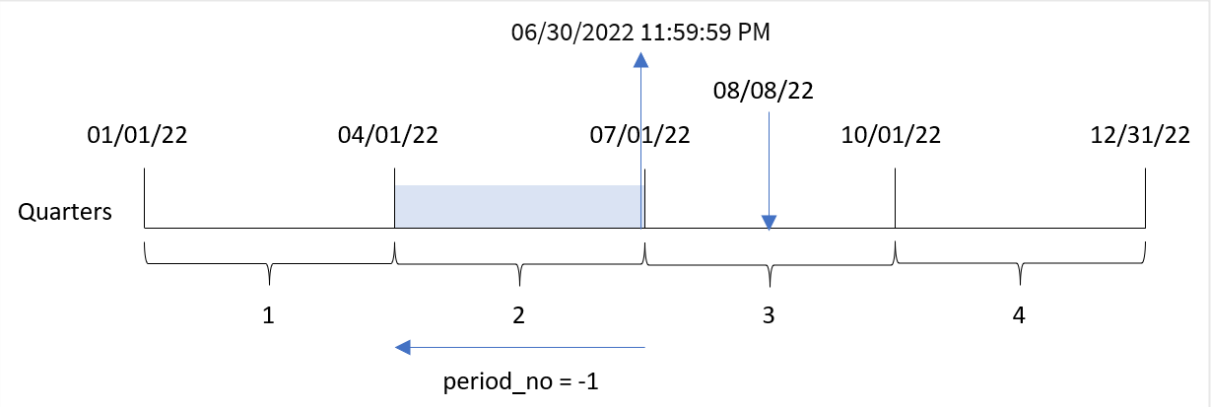
결과 테이블

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

id	date	previous_quarter_end	previous_quarter_end_timestamp
8198	7/9/2022	06/30/2022	6/30/2022 11:59:59 PM
8199	7/22/2022	06/30/2022	6/30/2022 11:59:59 PM
8200	7/23/2022	06/30/2022	6/30/2022 11:59:59 PM
8201	7/27/2022	06/30/2022	6/30/2022 11:59:59 PM
8202	8/2/2022	06/30/2022	6/30/2022 11:59:59 PM
8203	8/8/2022	06/30/2022	6/30/2022 11:59:59 PM
8204	8/19/2022	06/30/2022	6/30/2022 11:59:59 PM
8205	9/26/2022	06/30/2022	6/30/2022 11:59:59 PM
8206	10/14/2022	09/30/2022	9/30/2022 11:59:59 PM
8207	10/29/2022	09/30/2022	9/30/2022 11:59:59 PM

period_no -1은 quarterend() 함수에서 오프셋 인수로 사용되므로 함수는 먼저 트랜잭션이 발생한 분기를 식별합니다. 그런 다음 한 분기 이전으로 시프트하고 해당 분기의 마지막 밀리초를 식별합니다.

period_no가 -1인 quarterend() 함수의 다이어그램



트랜잭션 8203은 8월 8일에 발생했습니다. quarterend() 함수는 트랜잭션이 발생하기 전 분기가 4월 1일에서 6월 30일 사이임을 식별합니다. 그런 다음 함수는 해당 분기의 마지막 밀리초인 6월 30일 오후 11:59:59를 반환합니다.

예 3 - first_month_of_year

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 'Transactions'라는 테이블에 로드되는 2022년의 트랜잭션 집합을 포함하는 데이터 집합.
- 다음을 포함하는 선행 LOAD:
    - 'end_of_quarter' 필드로 설정되고 트랜잭션이 발생한 분기 끝의 타임스탬프를 반환하는 quarterend() 함수.
    - 'end_of_quarter_timestamp' 필드로 설정되어 선택한 분기의 끝 부분에 대한 정확한 타임스탬프를 반환하는 timestamp() 함수.

그러나 이 예에서 회사 정책은 회계 연도가 3월 1일에 시작하는 것입니다.

로드 스크립트

```
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
];
```

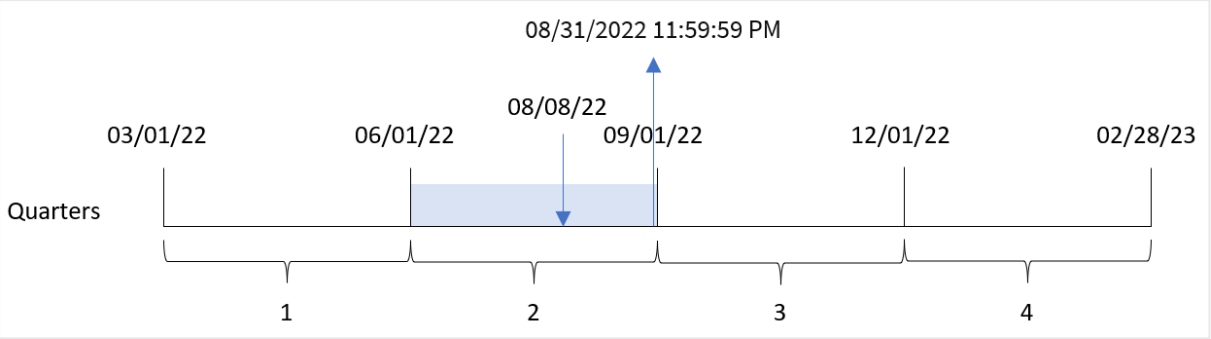
결과

결과 테이블

id	date	end_of_quarter	end_of_quarter_timestamp
8188	1/7/2022	02/28/2022	2/28/2022 11:59:59 PM
8189	1/19/2022	02/28/2022	2/28/2022 11:59:59 PM
8190	2/5/2022	02/28/2022	2/28/2022 11:59:59 PM
8191	2/28/2022	02/28/2022	2/28/2022 11:59:59 PM
8192	3/16/2022	05/31/2022	5/31/2022 11:59:59 PM
8193	4/1/2022	05/31/2022	5/31/2022 11:59:59 PM
8194	5/7/2022	05/31/2022	5/31/2022 11:59:59 PM
8195	5/16/2022	05/31/2022	5/31/2022 11:59:59 PM
8196	6/15/2022	08/31/2022	8/31/2022 11:59:59 PM
8197	6/26/2022	08/31/2022	8/31/2022 11:59:59 PM
8198	7/9/2022	08/31/2022	8/31/2022 11:59:59 PM
8199	7/22/2022	08/31/2022	8/31/2022 11:59:59 PM
8200	7/23/2022	08/31/2022	8/31/2022 11:59:59 PM
8201	7/27/2022	08/31/2022	8/31/2022 11:59:59 PM
8202	8/2/2022	08/31/2022	8/31/2022 11:59:59 PM
8203	8/8/2022	08/31/2022	8/31/2022 11:59:59 PM
8204	8/19/2022	08/31/2022	8/31/2022 11:59:59 PM
8205	9/26/2022	11/30/2022	11/30/2022 11:59:59 PM
8206	10/14/2022	11/30/2022	11/30/2022 11:59:59 PM
8207	10/29/2022	11/30/2022	11/30/2022 11:59:59 PM

`quarterend()` 함수에서 `first_month_of_year` 인수 `3`이 사용되므로 연도의 시작이 1월 1일에서 3월 1일로 이동합니다.

3월이 연도의 첫 번째 달인 quarterend() 함수의 다이어그램



트랜잭션 8203은 8월 8일에 발생했습니다. 연도의 시작이 3월 1일이기 때문에 해당 연도의 분기는 3월-5월, 6월-8월, 9월-11월 및 12월-2월 사이에 발생합니다.

quarterend() 함수는 트랜잭션이 6월 초와 8월 사이의 분기에 발생했음을 식별하고 해당 분기의 마지막 밀리초인 8월 31일 오후 11:59:59를 반환합니다.

예 4 - 차트 개체 예

로드 스크립트 및 차트 표현식

개요

첫 번째 예와 동일한 데이터 집합 및 시나리오가 사용됩니다.

그러나 이 예에서 데이터 집합은 변경되지 않고 응용 프로그램에 로드됩니다. 트랜잭션이 발생한 분기 말의 타임스탬프를 반환하는 계산은 앱의 차트에서 측정값으로 만들어집니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- id
- date

트랜잭션이 발생한 분기의 끝 날짜를 계산하려면 다음 측정값을 만듭니다.

- =quarterend(date)
- =timestamp(quarterend(date))

결과 테이블

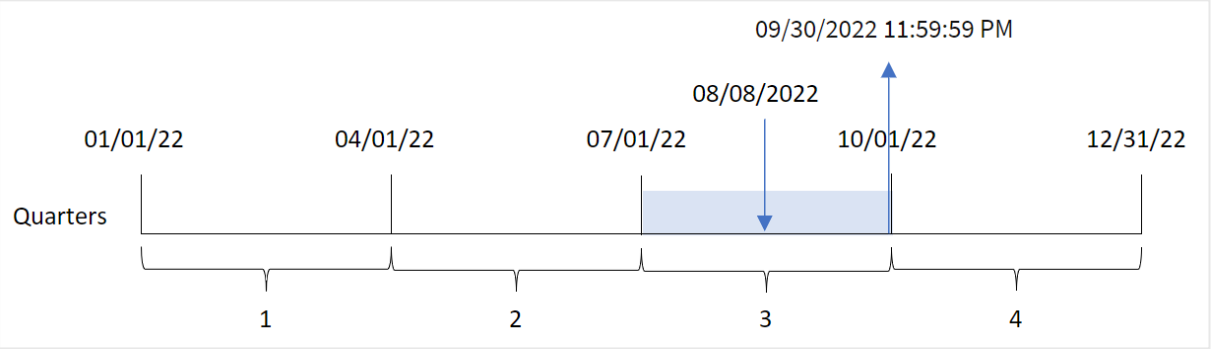
id	date	=quarterend(date)	=timestamp(quarterend(date))
8188	1/7/2022	03/31/2022	3/31/2022 11:59:59 PM
8189	1/19/2022	03/31/2022	3/31/2022 11:59:59 PM
8190	2/5/2022	03/31/2022	3/31/2022 11:59:59 PM
8191	2/28/2022	03/31/2022	3/31/2022 11:59:59 PM
8192	3/16/2022	03/31/2022	3/31/2022 11:59:59 PM
8193	4/1/2022	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

id	date	=quarterend(date)	=timestamp(quarterend(date))
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

'end_of_quarter' 필드는 quarterend() 함수를 사용하고 날짜 필드를 함수의 인수로 전달하여 선행 LOAD 문에서 만들어집니다.

quarterend() 함수는 초기에 날짜 값이 속하는 분기를 식별한 다음 해당 분기의 마지막 밀리초에 대한 타임스탬프를 반환합니다.

트랜잭션 8203의 분기 끝이 식별된 quarterend() 함수의 다이어그램



트랜잭션 8203은 8월 8일에 발생했습니다. quarterend() 함수는 트랜잭션이 3분기에 발생했음을 식별하고 해당 분기의 마지막 밀리초인 9월 30일 오후 11:59:59를 반환합니다.

예 5 - 시나리오

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 'Employee_Expenses'라는 테이블에 로드되는 데이터 집합. 테이블에는 다음 필드가 포함됩니다.
  - 직원 ID
  - 직원 이름
  - 각 직원의 평균 일일 비용 청구

최종 사용자는 직원 ID와 직원 이름별로 해당 분기에 남은 기간 동안 발생할 것으로 예상되는 비용 청구를 표시하는 차트 개체를 원합니다. 회계 연도는 1월에 시작됩니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- employee_id
- employee_name

누적 이자를 계산하려면 다음 측정값을 만듭니다.

- =(quarterend(today(1))-today(1))*avg_daily_claim

측정값의 **숫자 형식**을 **화폐**로 설정합니다.

결과 테이블

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

quarterend() 함수는 오늘 날짜를 유일한 인수로 사용하고 현재 월의 끝 날짜를 반환합니다. 그런 다음 연도 끝 날짜에서 오늘 날짜를 뺀 다음 이 표현식은 이번 달에 남아 있는 일수를 반환합니다.

그런 다음 이 값에 각 직원의 평균 일일 비용 청구를 곱하여 각 직원이 남은 분기에 예상되는 청구 금액을 계산합니다.

quartername

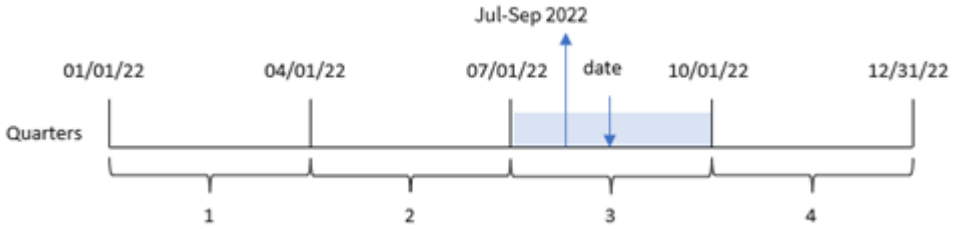
이 함수는 해당 분기의 첫 번째 날의 첫 번째 밀리초의 타임스탬프에 해당하는 기본 숫자 값으로 분기의 월(**MonthNames** 스크립트 변수에 따라 서식 지정) 및 연도를 보여주는 표시 값을 반환합니다.

구문:

```
QuarterName(date[, period_no[, first_month_of_year]])
```

반환 데이터 유형: dual

quartername() 함수 다이어그램



quartername() 함수는 날짜가 속하는 분기를 확인합니다. 그런 다음 이 분기의 시작-끝 월과 연도를 표시하는 값을 반환합니다. 이 결과의 기본 숫자 값은 분기의 첫 번째 밀리초입니다.

인수

인수	설명
date	평가할 날짜 또는 타임스탬프입니다.
period_no	**period_no**는 정수이며, 값 0은 **date**를 포함하는 분기를 나타냅니다. **period_no**가 음수 값일 경우 이전 분기, 양수 값일 경우 다음 분기를 나타냅니다.
first_month_of_year	1월에 시작되지 않는 (회계)연도를 사용하려는 경우 **first_month_of_year**에 2와 12 사이의 값을 지정하십시오.

사용 시기

quartername() 함수는 분기별로 집계를 비교하려는 경우에 유용합니다. 예를 들어 분기별 제품의 총 판매를 보려는 경우에 사용할 수 있습니다.

이 함수는 로드 스크립트에서 사용하여 마스터 캘린더 테이블에 필드를 만들 수 있습니다. 또는 차트에서 계산된 차원으로 직접 사용할 수 있습니다.

이 예에서는 날짜 서식 MM/DD/YYYY를 사용합니다. 날짜 서식은 데이터 로드 스크립트 맨 위에서 SET DateFormat 문으로 지정됩니다. 이 예의 형식을 필요에 따라 변경하십시오.

함수 예

예	결과
quartername('10/29/2013')	Oct-Dec 2013를 반환합니다.
quartername('10/29/2013', -1)	Jul-Sep 2013를 반환합니다.
quartername('10/29/2013', 0, 3)	Sep-Nov 2013을 반환합니다.

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

예 1 – 추가 인수가 없는 날짜

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Transactions라는 테이블에 로드되는 2022년 트랜잭션 집합이 포함된 데이터 집합.
- DateFormat 시스템 변수(MM/DD/YYYY) 서식으로 제공된 날짜 필드.
- 트랜잭션이 발생한 분기를 반환하는 필드 transaction_quarter 만들기.

필요에 따라 목록 등을 사용하여 여기에 다른 텍스트를 추가합니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- `date`
- `transaction_quarter`

결과 테이블

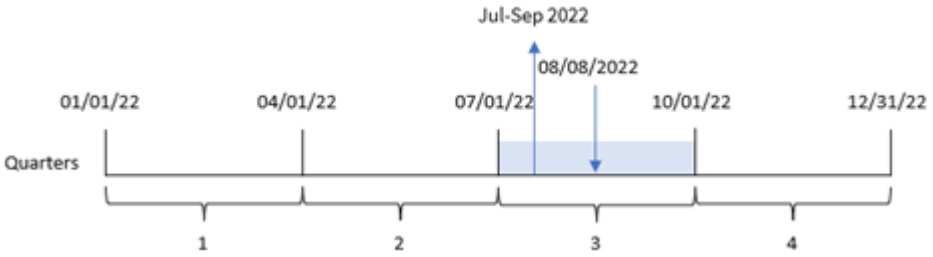
date	transaction_quarter
1/7/2022	Jan-Mar 2022
1/19/2022	Jan-Mar 2022
2/5/2022	Jan-Mar 2022
2/28/2022	Jan-Mar 2022
3/16/2022	Jan-Mar 2022
4/1/2022	Apr-Jun 2022
5/7/2022	Apr-Jun 2022
5/16/2022	Apr-Jun 2022
6/15/2022	Apr-Jun 2022
6/26/2022	Apr-Jun 2022
7/9/2022	Jul-Sep 2022
7/22/2022	Jul-Sep 2022
7/23/2022	Jul-Sep 2022
7/27/2022	Jul-Sep 2022
8/2/2022	Jul-Sep 2022
8/8/2022	Jul-Sep 2022
8/19/2022	Jul-Sep 2022

date	transaction_quarter
9/26/2022	Jul-Sep 2022
10/14/2022	Oct-Dec 2022
10/29/2022	Oct-Dec 2022

transaction_quarter 필드는 quartername() 함수를 사용하고 날짜 필드를 함수의 인수로 전달하여 선행 LOAD 문에서 만들어집니다.

quartername() 함수는 초기에 날짜 값이 속하는 분기를 식별합니다. 그런 다음 이 분기의 시작-끝 월과 연도를 표시하는 값을 반환합니다.

quartername() 함수 다이어그램, 추가 인수가 없는 예



트랜잭션 8203은 2022년 8월 8일에 발생했습니다. quartername() 함수는 트랜잭션이 3분기에 발생했음을 식별하므로 2022년 7월-9월을 반환합니다. MonthNames 시스템 변수와 동일한 서식으로 월이 표시됩니다.

예 2 – period_no 인수를 사용한 날짜

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 첫 번째 예와 동일한 데이터 집합 및 시나리오.
- 이전 분기를 트랜잭션이 발생한 시점으로 되돌리는 필드 previous_quarter 만들기.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- `date`
- `previous_quarter`

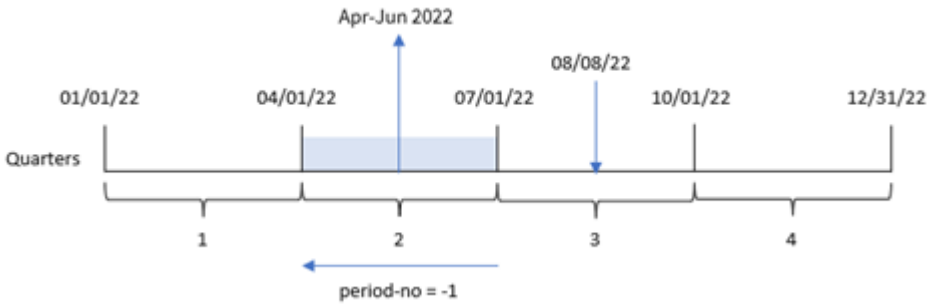
결과 테이블

date	previous_quarter
1/7/2022	Oct-Dec 2021
1/19/2022	Oct-Dec 2021
2/5/2022	Oct-Dec 2021
2/28/2022	Oct-Dec 2021
3/16/2022	Oct-Dec 2021
4/1/2022	Jan-Mar 2022
5/7/2022	Jan-Mar 2022
5/16/2022	Jan-Mar 2022
6/15/2022	Jan-Mar 2022
6/26/2022	Jan-Mar 2022

date	previous_quarter
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

이 경우 quartername() 함수에서 period_no -1을 오프셋 인수로 사용했으므로 이 함수는 먼저 트랜잭션이 3분기에 발생했음을 식별합니다. 그런 다음 한 분기 이전으로 시프트하고 이 분기의 시작-끝 월과 연도를 표시하는 값을 반환합니다.

quartername() 함수의 다이어그램, period_no 예



트랜잭션 8203은 8월 8일에 발생했습니다. quartername() 함수는 트랜잭션이 발생하기 전 분기가 4월 1일에서 6월 30일 사이임을 식별합니다. 따라서 2022년 4월-6월을 반환합니다.

예 3 – first_week_day 인수를 사용한 날짜

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

이 로드 스크립트에는 첫 번째 예와 동일한 데이터 집합 및 시나리오가 포함되어 있습니다. 그러나 이 예에서는 회계 연도의 시작으로 3월 1일을 설정해야 합니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- transaction_quarter

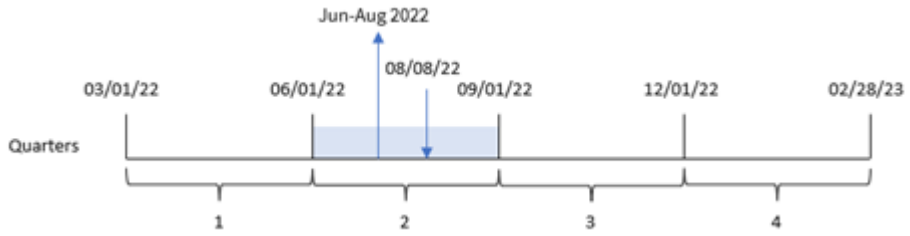
결과 테이블

date	transaction_quarter
1/7/2022	Dec-Feb 2021
1/19/2022	Dec-Feb 2021
2/5/2022	Dec-Feb 2021

date	transaction_quarter
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

이 경우 quartername() 함수에서 first_month_of_year 인수 3을 사용하므로 연도의 시작이 1월 1일에서 3월 1일로 이동합니다. 따라서 1년의 분기는 3월-5월, 6월-8월, 9월-11월 및 12월-2월로 구분됩니다.

quartername() 함수 다이어그램, first_week_day 예



트랜잭션 8203은 8월 8일에 발생했습니다. quartername() 함수는 트랜잭션이 6월 초에서 8월 말 사이의 2분기에 발생했음을 식별합니다. 따라서 2022년 6월-8월을 반환합니다.

예 4 – 차트 개체 예

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

이 로드 스크립트에는 첫 번째 예와 동일한 데이터 집합 및 시나리오가 포함되어 있습니다.

그러나 이 예에서는 변경되지 않은 데이터 집합이 응용 프로그램에 로드됩니다. 트랜잭션이 발생한 분기의 끝에 대한 타임스탬프를 반환하는 계산은 응용 프로그램의 차트 개체에서 측정값으로 만들어집니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다. `date`.

다음 측정값을 만듭니다.

```
=quartername(date)
```

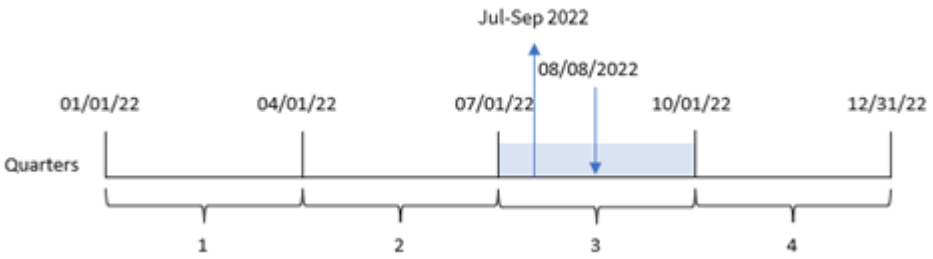
결과 테이블

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

`transaction_quarter` 측정값은 `quartername()` 함수를 사용하고 `date` 필드를 함수의 인수로 전달하여 차트 개체에서 만들어집니다.

`quartername()` 함수는 초기에 날짜 값이 속하는 분기를 식별합니다. 그런 다음 이 분기의 시작-끝 월과 연도를 표시하는 값을 반환합니다.

quartername() 함수의 다이어그램, 차트 개체 예



트랜잭션 8203은 2022년 8월 8일에 발생했습니다. quartername() 함수는 트랜잭션이 3분기에 발생했음을 식별하므로 2022년 7월-9월을 반환합니다. MonthNames 시스템 변수와 동일한 서식으로 월이 표시됩니다.

예 5 – 시나리오

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Transactions라는 테이블에 로드되는 2022년 트랜잭션 집합이 포함된 데이터 집합.
- DateFormat 시스템 변수(MM/DD/YYYY) 서식으로 제공된 날짜 필드.

최종 사용자는 트랜잭션에 대한 분기별 총 판매를 나타내는 차트 개체를 원합니다. 데이터 모델에서 이 차원을 사용할 수 없는 경우에도 quartername() 함수를 차트에서 계산된 차원으로 사용하여 이를 달성할 수 있습니다.

로드 스크립트

```
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
];
```

결과

다음과 같이 하십시오.

1. 데이터를 로드하고 시트를 엽니다. 새 테이블을 만듭니다.
2. 다음 표현식을 사용하여 계산된 차원을 만듭니다.
   `=quartername(date)`
3. 그런 후, 다음 집계 측정값을 사용하여 총 판매액을 계산합니다.
   `=sum(amount)`
4. 측정값의 **숫자 형식**을 **화폐**로 설정합니다.

결과 테이블

=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

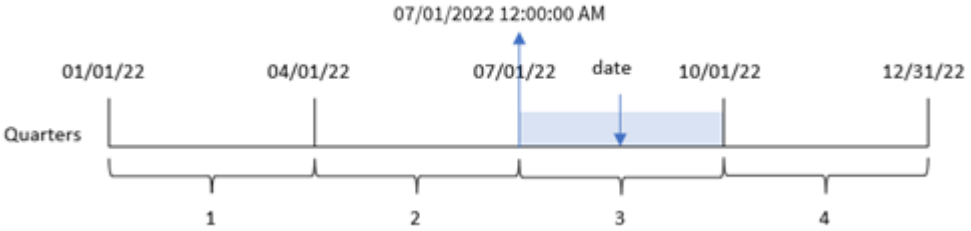
이 함수는 **date**를 포함하는 분기의 첫 번째 밀리초의 타임스탬프에 해당하는 값을 반환합니다. 기본 출력 형식은 스크립트에 설정된 **DateFormat**입니다.

구문:

```
QuarterStart(date[, period_no[, first_month_of_year]])
```

반환 데이터 유형: dual

quarterstart() 함수 다이어그램



quarterstart() 함수는 date가 속하는 분기를 확인합니다. 그런 다음 해당 분기의 첫 번째 달의 첫 번째 밀리초에 대한 타임스탬프를 날짜 서식으로 반환합니다.

인수

인수	설명
date	평가할 날짜 또는 타임스탬프입니다.
period_no	**period_no**는 정수이며, 값 0은 **date**를 포함하는 분기를 나타냅니다. **period_no**가 음수 값일 경우 이전 분기, 양수 값일 경우 다음 분기를 나타냅니다.
first_month_of_year	1월에 시작되지 않는 (회계)연도를 사용하려는 경우 **first_month_of_year**에 2와 12 사이의 값을 지정하십시오.

사용 시기

quarterstart() 함수는 일반적으로 사용자가 지금까지 경과된 분기의 부분을 사용하여 계산하려고 할 때 표현식의 일부로 사용됩니다. 예를 들어, 사용자가 해당 분기부터 현재까지 누적된 이자를 계산하려는 경우 사용할 수 있습니다.

함수 예

예	결과
quarterstart('10/29/2005')	10/01/2005를 반환합니다.
quarterstart('10/29/2005', -1)	07/01/2005를 반환합니다.
quarterstart('10/29/2005', 0, 3)	09/01/2005를 반환합니다.

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

예 1 – 추가 인수 없음

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- `Transactions`라는 테이블에 로드되는 2022년 트랜잭션 집합이 포함된 데이터 집합.
- `DateFormat` 시스템 변수(MM/DD/YYYY) 서식으로 제공된 날짜 필드.
- 트랜잭션이 발생한 분기 시작에 대한 타임스탬프를 반환하는 필드 `start_of_quarter` 만들기.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

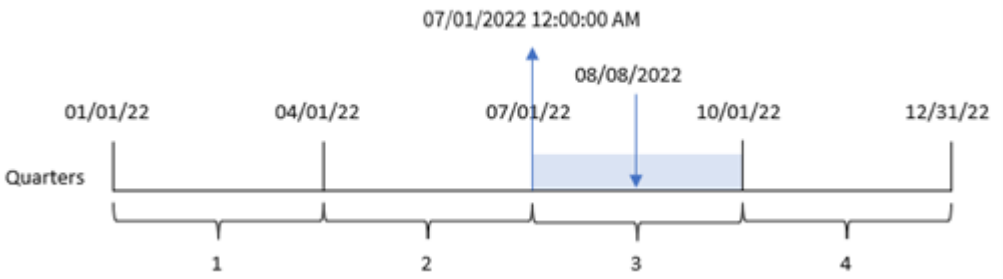
- date
- start_of_quarter
- start_of_quarter_timestamp

결과 테이블

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

start_of_quarter 필드는 quarterstart() 함수를 사용하고 날짜 필드를 함수의 인수로 전달하여 선행 LOAD 문에서 만들어집니다. quarterstart() 함수는 초기에 날짜 값이 속하는 분기를 식별합니다. 그런 다음 해당 분기의 첫 번째 밀리초에 대한 타임스탬프를 반환합니다.

quarterstart() 함수 다이어그램, 추가 인수가 없는 예



트랜잭션 8203은 8월 8일에 발생했습니다. quarterstart() 함수는 트랜잭션이 3분기에 발생했음을 식별하고 해당 분기의 첫 번째 밀리초인 7월 1일 오전 12:00:00를 반환합니다.

예 2 – period_no

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 첫 번째 예와 동일한 데이터 집합 및 시나리오.
- 트랜잭션이 발생하기 전 분기 시작에 대한 타임스탬프를 반환하는 필드 previous_quarter_start 만들기.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- previous_quarter_start
- previous_quarter_start_timestamp

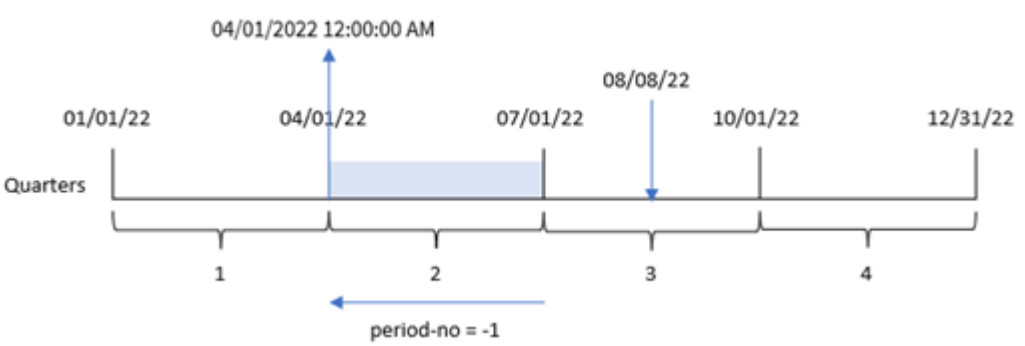
결과 테이블

date	previous_quarter_start	previous_quarter_start_timestamp
1/7/2022	10/01/2021	10/1/2021 12:00:00 AM
1/19/2022	10/01/2021	10/1/2021 12:00:00 AM
2/5/2022	10/01/2021	10/1/2021 12:00:00 AM
2/28/2022	10/01/2021	10/1/2021 12:00:00 AM
3/16/2022	10/01/2021	10/1/2021 12:00:00 AM
4/1/2022	01/01/2022	1/1/2022 12:00:00 AM
5/7/2022	01/01/2022	1/1/2022 12:00:00 AM
5/16/2022	01/01/2022	1/1/2022 12:00:00 AM
6/15/2022	01/01/2022	1/1/2022 12:00:00 AM
6/26/2022	01/01/2022	1/1/2022 12:00:00 AM
7/9/2022	04/01/2022	4/1/2021 12:00:00 AM
7/22/2022	04/01/2022	4/1/2021 12:00:00 AM
7/23/2022	04/01/2022	4/1/2021 12:00:00 AM
7/27/2022	04/01/2022	4/1/2021 12:00:00 AM

date	previous_quarter_start	previous_quarter_start_timestamp
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

이 경우 quarterstart() 함수에서 period_no -1을 오프셋 인수로 사용했으므로 이 함수는 먼저 트랜잭션이 발생한 분기를 식별합니다. 그런 다음 한 분기 이전으로 시프트하고 해당 분기의 첫 번째 밀리초를 식별합니다.

quarterstart() 함수의 다이어그램, period_no 예



트랜잭션 8203은 8월 8일에 발생했습니다. quarterstart() 함수는 트랜잭션이 발생하기 전 분기가 4월 1일에서 6월 30일 사이임을 식별합니다. 그런 다음 해당 분기의 첫 번째 밀리초, 즉 4월 1일 오전 12:00:00를 반환합니다.

예 3 – first_month_of_year

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

이 로드 스크립트에는 첫 번째 예와 동일한 데이터 집합 및 시나리오가 포함되어 있습니다. 그러나 이 예에서는 회계 연도의 시작으로 3월 1일을 설정해야 합니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- start_of_quarter
- start_of_quarter_timestamp

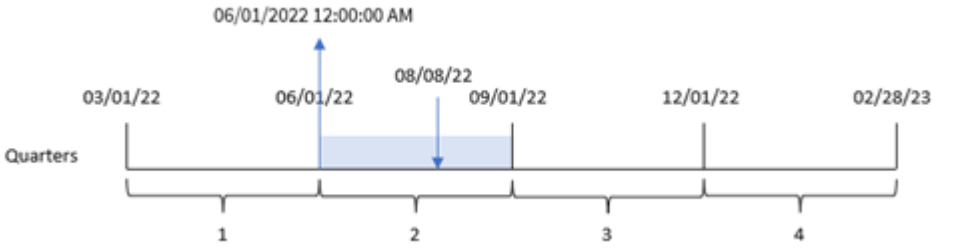
결과 테이블

date	start_of_quarter	start_of_quarter_timestamp
1/7/2022	12/01/2021	12/1/2021 12:00:00 AM
1/19/2022	12/01/2021	12/1/2021 12:00:00 AM
2/5/2022	12/01/2021	12/1/2021 12:00:00 AM
2/28/2022	12/01/2021	12/1/2021 12:00:00 AM
3/16/2022	03/01/2022	3/1/2022 12:00:00 AM
4/1/2022	03/01/2022	3/1/2022 12:00:00 AM

date	start_of_quarter	start_of_quarter_timestamp
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

이 경우 quarterstart()함수에서 first_month_of_year 인수 3이 사용되므로 연도의 시작이 1월 1일에서 3월 1일로 이동합니다.

quarterstart() 함수의 다이어그램, first_month_of_year 예



트랜잭션 8203은 8월 8일에 발생했습니다. 연도의 시작이 3월 1일이므로 해당 연도의 분기는 3월-5월, 6월-8월, 9월-11월 및 12월-2월 사이에 발생합니다. quarterstart() 함수는 트랜잭션이 6월 초와 8월 사이의 분기에 발생했음을 식별하고 해당 분기의 첫 번째 밀리초인 6월 1일 오전 12:00:00를 반환합니다.

예 4 – 차트 개체 예

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

이 로드 스크립트에는 첫 번째 예와 동일한 데이터 집합 및 시나리오가 포함되어 있습니다.

그러나 이 예에서는 변경되지 않은 데이터 집합이 응용 프로그램에 로드됩니다. 트랜잭션이 발생한 분기의 끝에 대한 타임스탬프를 반환하는 계산은 응용 프로그램의 차트 개체에서 측정값으로 만들어집니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다. `date`.

다음 측정값을 추가합니다.

- `=quarterstart(date)`
- `=timestamp(quarterstart(date))`

결과 테이블

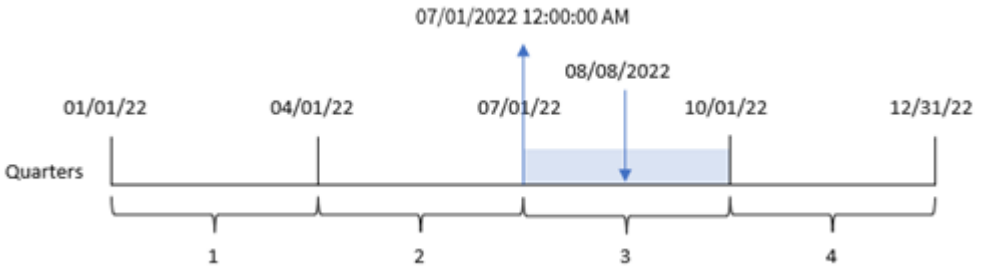
date	=quarterstart(date)	=timestamp(quarterstart(date))
10/14/2022	10/01/2022	10/1/2022 12:00:00 AM
10/29/2022	10/01/2022	10/1/2022 12:00:00 AM
7/9/2022	07/01/2022	7/1/2022 12:00:00 AM

date	=quarterstart(date)	=timestamp(quarterstart(date))
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

start_of_quarter 측정값은 quarterstart() 함수를 사용하고 date 필드를 함수의 인수로 전달하여 차트 개체에서 만들어집니다.

quarterstart() 함수는 날짜 값이 속하는 분기를 식별하여 해당 분기의 첫 번째 밀리초에 대한 타임스탬프를 반환합니다.

quarterstart() 함수의 다이어그램, 차트 개체 예



트랜잭션 8203은 8월 8일에 발생했습니다. quarterstart() 함수는 트랜잭션이 3분기에 발생했음을 식별하고 해당 분기의 첫 번째 밀리초를 반환합니다. 이 반환된 값은 7월 1일 오전 12:00:00입니다.

예 5 – 시나리오

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Loans라는 테이블에 로드되는 대출 잔액 집합이 포함된 데이터 집합.
- 대출 ID, 분기 초 잔액 및 각 대출에 연간 부과되는 단순 이자율로 구성된 데이터.

최종 사용자는 해당 분기부터 현재까지 각 대출에 대해 발생한 현재 이자를 대출 ID별로 표시하는 차트 개체를 원합니다.

로드 스크립트

```
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
];
```

결과

다음과 같이 하십시오.

1. 데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.
   - loan_id
   - start_balance
2. 다음으로 누적 이자를 계산하는 이 측정값을 만듭니다.
   `=start_balance*(rate*(today(1)-quarterstart(today(1)))/365)`
3. 측정값의 **숫자 형식**을 **화폐**로 설정합니다.

결과 테이블

loan_id	start_balance	=start_balance*(rate*(today(1)-quarterstart(today(1)))/365)
8188	$10000.00	$15.07
8189	$15000.00	$128.84

loan_id	start_balance	=start_balance*(rate*(today(1)-quarterstart(today(1)))/365)
8190	$17500.00	$63.29
8191	$21000.00	$107.59
8192	$90000.00	$1139.18

오늘 날짜를 유일한 인수로 사용하는 quarterstart() 함수는 현재 연도의 시작 날짜를 반환합니다. 이 표현식은 현재 날짜에서 해당 결과를 빼서 이번 분기에 지금까지 경과한 일수를 반환합니다.

그런 다음 이 값에 이자율을 곱하고 365로 나누어 이 기간 동안 발생한 유효 이자율을 반환합니다. 그런 다음 결과에 대출의 시작 잔액을 곱하여 이번 분기에 지금까지 발생한 이자를 반환합니다.

second

이 함수는 **expression**의 분위수가 표준 숫자 해석에 따라 시간으로 해석될 경우 초를 나타내는 정수를 반환합니다.

구문:

```
second (expression)
```

반환 데이터 유형: 정수

사용 시기

second() 함수는 집계를 초 단위로 비교하려는 경우에 유용합니다. 예를 들어, 초 단위로 활동 횟수 분포를 보고 싶을 때 이 함수를 사용할 수 있습니다.

이러한 차원은 마스터 캘린더 테이블에 필드를 만드는 함수를 사용하여 로드 스크립트에서 만들거나 차트에서 계산된 차원으로 직접 사용할 수 있습니다.

함수 예

예	결과
second('09:14:36')	36를 반환합니다.
second('0.5555')	0.5555 = 13:19:55이므로, 55를 반환합니다.

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

예 1 – 변수

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Transactions라는 테이블에 로드되는 타임스탬프별 트랜잭션을 포함하는 데이터 집합.
- 기본 TimeStamp 시스템 변수(M/D/YYYY h:mm:ss[.fff] TT)가 사용됩니다.
- 구매가 발생하는 시점을 계산하는 필드(second) 만들기.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- second

결과 테이블

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

second 필드의 값은 second() 함수를 사용하고 선행 LOAD 문의 표현식으로 날짜를 전달하여 만들어집니다.

예 2 – 차트 개체

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

이 로드 스크립트에는 첫 번째 예와 동일한 데이터 집합 및 시나리오가 포함되어 있습니다. 그러나 이 예에서는 변경되지 않은 데이터 집합이 응용 프로그램에 로드됩니다. second 값은 차트 개체의 측정값을 통해 계산됩니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다.date.

다음 측정값을 만듭니다.

```
=second(date)
```

결과 테이블

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

second의 값은 second() 함수를 사용하고 차트 개체에 대한 측정값의 표현식으로 날짜를 전달하여 만들어집니다.

예 3 – 시나리오

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 특정 축제의 티켓 판매 웹사이트에 대한 트래픽을 나타내기 위해 생성되는 타임스탬프 데이터 집합. 이러한 타임스탬프와 해당id는 Web_Traffic이라는 테이블에 로드됩니다.
- TimeStamp 시스템 변수 M/D/YYYY h:mm:ss[.fff] TT가 사용됩니다.

이 시나리오에서는 10,000개의 티켓이 있었고 2021년 5월 20일 오전 9시에 판매가 시작되었습니다. 1분 만에 티켓이 매진되었습니다.

사용자는 웹사이트 방문 횟수를 초 단위로 보여 주는 차트 개체를 원합니다.

로드 스크립트

```
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;
```

결과

다음과 같이 하십시오.

1. 데이터를 로드하고 시트를 엽니다. 새 테이블을 만듭니다.
2. 다음으로 다음 표현식을 사용하여 계산된 차원을 만듭니다.
   `=second(timestamp)`
3. 총 항목 수를 계산하는 집계 측정값을 만듭니다.
   `=count(id)`

결과 테이블은 아래 테이블과 유사하지만 집계 측정값에 대한 값은 다릅니다.

결과 테이블

second(timestamp)	=count(id)
0	150
1	184
2	163
3	178
4	179
5	158
6	177
7	169

second(timestamp)	=count(id)
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
+ 39개 추가 행	

setdateyear

이 함수는 **timestamp** 및 **year** 를 입력으로 사용하여 입력에 지정된 **year** 로 **timestamp** 를 업데이트합니다.

구문:

```
setdateyear (timestamp, year)
```

반환 데이터 유형: dual

인수:

인수

인수	설명
timestamp	표준 Qlik Sense 타임스탬프(대개의 경우 단순히 날짜)입니다.
year	네 자리의 연도입니다.

예 및 결과:
이 예에서는 날짜 서식 **DD/MM/YYYY**를 사용합니다. 날짜 서식은 데이터 로드 스크립트 맨 위에서 **SET DateFormat** 문으로 지정됩니다. 이 예제의 서식을 필요에 따라 변경하십시오.

스크립팅 예

예	결과
setdateyear ('29/10/2005', 2013)	'29/10/2013을 반환합니다.
setdateyear ('29/10/2005 04:26:14', 2013)	'29/10/2013 04:26:14'를 반환합니다. 시각화의 타임스탬프에서 시간 부분을 표시하려면 숫자 서식을 날짜로 설정하고 시간 값을 표시하는 서식 값을 선택해야 합니다.

예제 스크립트를 앱에 추가하고 실행합니다. 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.

```
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

];
```

결과 테이블에는 원래 날짜와 연도가 2013으로 설정된 열이 포함됩니다.

결과 테이블

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

이 함수는 **timestamp**, **month** 및 **year** 를 입력으로 사용하여 입력에 지정된 **year**및 **month** 로 **timestamp** 를 업데이트합니다. .

구문:

```
SetDateYearMonth (timestamp, year, month)
```

반환 데이터 유형: dual

인수:

인수

인수	설명
timestamp	표준 Qlik Sense 타임스탬프(대개의 경우 단순히 날짜)입니다.
year	네 자리의 연도입니다.
month	하나 또는 두 자리의 월입니다.

예 및 결과:
이 예에서는 날짜 서식 **DD/MM/YYYY**를 사용합니다. 날짜 서식은 데이터 로드 스크립트 맨 위에서 **SET DateFormat** 문으로 지정됩니다. 이 예제의 서식을 필요에 따라 변경하십시오.

스크립팅 예

예	결과
setdateyearmonth ('29/10/2005', 2013, 3)	'29/03/2013을 반환합니다.
setdateyearmonth ('29/10/2005 04:26:14', 2013, 3)	'29/03/2013 04:26:14'를 반환합니다. 시각화의 타임스탬프에서 시간 부분을 표시하려면 숫자 서식을 날짜로 설정하고 시간 값을 표시하는 서식 값을 선택해야 합니다.

예제 스크립트를 앱에 추가하고 실행합니다. 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.

```
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

];
```

결과 테이블에는 원래 날짜와 연도가 2013으로 설정된 열이 포함됩니다.

결과 테이블

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

이 함수는 Qlik 엔진이 실행 중인 컴퓨터에 정의된 표준 시간대를 반환합니다.

구문:

```
TimeZone( )
```

반환 데이터 유형: dual

`timezone( )`
앱의 측정값에서 다른 표준 시간대를 보려면 측정값에서 `localtime()` 함수를 사용할 수 있습니다.

today

이 함수는 현재 날짜를 반환합니다. 이 함수는 `DateFormat` 시스템 변수 서식으로 값을 반환합니다.

구문:

```
today([ timer_mode])
```

반환 데이터 유형: dual

`today()` 함수는 로드 스크립트 또는 차트 개체에서 사용할 수 있습니다.

기본 `timer_mode` 값은 1입니다.

인수

인수	설명
timer_mode	다음과 같은 값을 가질 수 있습니다. 0(데이터 로드가 마지막으로 완료된 날짜) 1(함수 호출 날짜) 2(앱이 열린 날짜) *로드 스크립트에서 이 함수를 사용하면 **timer_mode=0**이 마지막으로 데이터 로드가 완료된 날짜가 되지만 **timer_mode=1**은 현재 데이터 로드 날짜를 제공합니다.*

함수 예

timer_mode 값	로드 스크립트에서 사용된 경우 결과	차트 개체에서 사용된 경우 결과
0	최신 데이터 다시 로드 이전에 마지막으로 성공한 데이터 다시 로드의 날짜를 DateFormat 시스템 변수 형식으로 반환합니다.	최신 데이터를 다시 로드한 날짜를 DateFormat 시스템 변수 형식으로 반환합니다.
1	최신 데이터를 다시 로드한 날짜를 DateFormat 시스템 변수 형식으로 반환합니다.	함수 호출의 날짜를 DateFormat 시스템 변수 형식으로 반환합니다.
2	응용 프로그램에서 사용자의 세션이 시작된 날짜를 DateFormat 시스템 변수 형식으로 반환합니다. 사용자가 스크립트를 다시 로드하지 않으면 업데이트되지 않습니다.	응용 프로그램에서 사용자의 세션이 시작된 날짜를 DateFormat 시스템 변수 형식으로 반환합니다. 이는 새 세션이 시작되거나 응용 프로그램의 데이터가 다시 로드되면 새로 고쳐집니다.

사용 시기

today() 함수는 일반적으로 표현식 내에서 구성 요소로 사용됩니다. 예를 들어, 현재 날짜까지 한 달 동안 누적된 이자를 계산하는 데 사용할 수 있습니다.

다음 표는 timer_mode 인수에 대해 다른 값이 주어졌을 때 today() 함수에서 반환된 결과에 대한 설명을 제공합니다.

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

예 1 – 로드 스크립트를 사용한 개체 생성

로드 스크립트 및 결과

개요

다음 예에서는 today() 함수를 사용하여 세 개의 변수를 만듭니다. 각 변수는 timer_mode 옵션 중 하나를 사용하여 효과를 보여 줍니다.

변수가 목적을 나타내려면 스크립트를 다시 로드한 다음 24시간 후에 스크립트를 두 번째로 다시 로드합니다. 그러면 today(0) 및 today(1) 변수가 다른 값을 표시하여 목적을 올바르게 보여 줍니다.

로드 스크립트

```
LET vPreviousDataLoad = today(0);
LET vCurrentDataLoad = today(1);
LET vApplicationOpened = today(2);
```

결과

데이터가 두 번째로 로드되면 아래 지침을 사용하여 3개의 텍스트 상자를 만듭니다.

먼저 이전에 로드된 데이터에 대한 텍스트 상자를 만듭니다.

다음과 같이 하십시오.

1. **텍스트 및 이미지** 차트 개체를 사용하여 텍스트 상자를 만듭니다.
2. 개체에 다음 측정값을 추가합니다.
   `=vPreviousDataLoad`
3. **모양** 아래에서 **Show titles**를 선택하고 '이전 다시 로드 시간'이라는 제목을 개체에 추가합니다.

다음으로 현재 로드 중인 데이터에 대한 텍스트 상자를 만듭니다.

다음과 같이 하십시오.

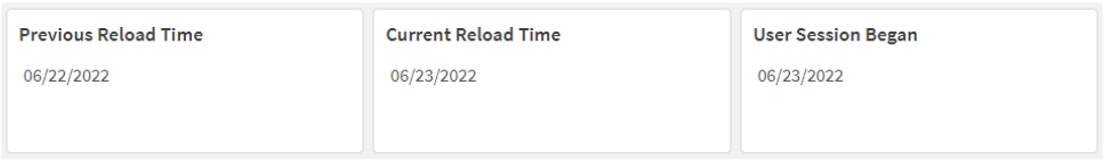
1. **텍스트 및 이미지** 차트 개체를 사용하여 텍스트 상자를 만듭니다.
2. 개체에 다음 측정값을 추가합니다.
   `=vCurrentDataLoad`
3. **모양** 아래에서 **Show titles**를 선택하고 '현재 다시 로드 시간'이라는 제목을 개체에 추가합니다.

응용 프로그램에서 사용자 세션이 시작된 시간을 표시할 최종 텍스트 상자를 만듭니다.

다음과 같이 하십시오.

1. **텍스트 및 이미지** 차트 개체를 사용하여 텍스트 상자를 만듭니다.
2. 개체에 다음 측정값을 추가합니다.
   `=vApplicationOpened`
3. **모양** 아래에서 **Show titles**를 선택하고 '사용자 세션 시작됨'이라는 제목을 개체에 추가합니다.

로드 스크립트에서 today() 함수를 사용하여 만들어진 변수 다이어그램



위 이미지는 만들어진 각 변수 값에 대한 예를 보여 줍니다. 예를 들어 값은 다음과 같을 수 있습니다.

- 이전 다시 로드 시간: 06/22/2022
- 현재 다시 로드 시간: 06/23/2022
- 사용자 세션 시작: 06/23/2022

예 2 – 로드 스크립트 없이 개체 생성

로드 스크립트 및 차트 표현식

개요

다음 예에서는 today() 함수를 사용하여 세 개의 차트 개체를 만듭니다. 각 차트 개체는 timer_mode 옵션 중 하나를 사용하여 효과를 보여 줍니다.

이 예에는 로드 스크립트가 없습니다.

결과

데이터가 두 번째로 로드되면 세 개의 텍스트 상자를 만듭니다.

먼저 최신 데이터 다시 로드를 위한 텍스트 상자를 만듭니다.

다음과 같이 하십시오.

1. **텍스트 및 이미지** 차트 개체를 사용하여 텍스트 상자를 만듭니다.
2. 다음 측정값을 추가합니다.
   `=today(0)`
3. **모양**에서 **제목 표시**를 선택하고 개체에 '최신 데이터 다시 로드'라는 제목을 추가합니다.

다음으로 현재 시간을 표시하는 텍스트 상자를 만듭니다.

다음과 같이 하십시오.

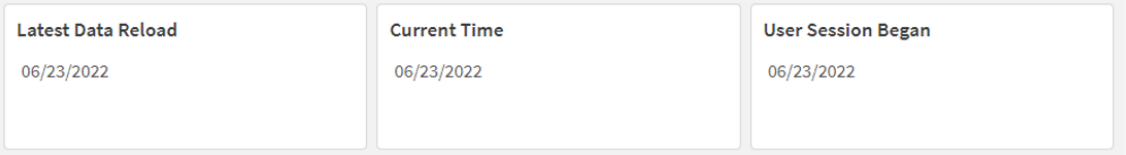
1. **텍스트 및 이미지** 차트 개체를 사용하여 텍스트 상자를 만듭니다.
2. 다음 측정값을 추가합니다.
   `=today(1)`
3. **모양**에서 **제목 표시**를 선택하고 개체에 '현재 시간'이라는 제목을 추가합니다.

응용 프로그램에서 사용자 세션이 시작된 시간을 표시할 최종 텍스트 상자를 만듭니다.

다음과 같이 하십시오.

1. **텍스트 및 이미지** 차트 개체를 사용하여 텍스트 상자를 만듭니다.
2. 다음 측정값을 추가합니다.
   `=today(2)`
3. **모양**에서 **제목 표시**를 선택하고 개체에 '사용자 세션 시작'이라는 제목을 추가합니다.

로드 스크립트 없이 today() 함수를 사용하여 만들어진 개체의 다이어그램



위 이미지는 만들어진 각 개체에 대한 예시 값을 보여 줍니다. 예를 들어 값은 다음과 같을 수 있습니다.

- 최신 데이터 다시 로드: 06/23/2022
- 현재 시간: 06/23/2022
- 사용자 세션 시작: 06/23/2022

'최신 데이터 다시 로드' 차트 개체는 `timer_mode` 값 `0`을 사용합니다. 이는 데이터가 성공적으로 다시 로드된 마지막 시간의 타임스탬프를 반환합니다.

'현재 시간' 차트 개체는 `timer_mode` 값 `1`을 사용합니다. 이는 시스템 시계에 따라 현재 시간을 반환합니다. 시트 또는 개체를 새로 고치면 이 값이 업데이트됩니다.

'사용자 세션 시작' 차트 개체는 `timer_mode` 값 `2`를 사용합니다. 이는 응용 프로그램이 열리고 사용자의 세션이 시작된 시간에 대한 타임스탬프를 반환합니다.

예 3 – 시나리오

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- `Loans`라는 테이블에 로드되는 대출 잔액 집합이 포함된 데이터 집합.
- 대출 ID, 월초 잔액, 각 대출에 연간 부과되는 단순 이자율에 대한 필드가 있는 테이블 데이터입니다.

최종 사용자는 해당 월부터 현재까지 각 대출에 대해 발생한 현재 이자를 대출 ID별로 표시하는 차트 개체를 원합니다. 응용 프로그램은 일주일에 한 번만 다시 로드되지만 사용자는 개체 또는 응용 프로그램이 새로 고쳐질 때마다 결과가 새로 고쳐지기를 원합니다.

로드 스크립트

```
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
];
```

결과

다음과 같이 하십시오.

1. 데이터를 로드하고 시트를 엽니다. 새 테이블을 만듭니다.
2. 다음 필드를 차원으로 추가합니다.
    - loan_id
    - start_balance
3. 다음으로 누적 이자를 계산하는 측정값을 만듭니다.
   `=start_balance*(rate*(today(1)-monthstart(today(1)))/365)`
4. 측정값의 **숫자 형식**을 **화폐**로 설정합니다.

결과 테이블

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

today() 함수를 사용하여 오늘 날짜를 유일한 인수로 반환하는 monthstart() 함수는 현재 월의 시작 날짜를 반환합니다. today() 함수를 다시 사용하여 현재 날짜에서 해당 결과를 빼면 이 표현식은 이번 달에 지금까지 경과한 일 수를 반환합니다.

그런 다음 이 값에 이자율을 곱하고 365로 나누어 이 기간 동안 발생한 유효 이자율을 반환합니다. 그런 다음 결과에 대출의 시작 잔액을 곱하여 이번 달에 지금까지 발생한 이자를 반환합니다.

표현식 내부의 today() 함수에서 값 1이 timer_mode 인수로 사용되므로 차트 개체를 새로 고칠 때마다(응용 프로그램 열기, 페이지 새로 고침, 시트 간 이동 등) 반환된 날짜는 현재 날짜이며 그에 따라 결과가 새로 고쳐집니다.

UTC

현재 Coordinated Universal Time을 반환합니다.

구문:

```
UTC( )
```

반환 데이터 유형: dual

```
utc( )
```

week

이 함수는 입력한 날짜에 해당하는 주차를 나타내는 정수를 반환합니다.

구문:

```
week(timestamp [, first_week_day [, broken_weeks [, reference_day]]])
```

반환 데이터 유형: 정수

인수

인수	설명
timestamp	평가할 날짜 또는 타임스탬프입니다.
first_week_day	주의 시작 요일을 지정합니다. 생략하면 변수 **FirstWeekDay**의 값이 사용됩니다. **first_week_day**에 가능한 값은 월요일에 0, 화요일에 1, 수요일에 2, 목요일에 3, 금요일에 4, 토요일에 5, 일요일에 6입니다. 시스템 변수에 대한 자세한 내용은 *FirstWeekDay (page 219)*를 참조하십시오.
broken_weeks	**broken_weeks**를 지정하지 않으면 **BrokenWeeks** 변수의 값이 주를 분리할지 여부를 정의하는 데 사용됩니다.
reference_day	**reference_day**를 지정하지 않으면 **ReferenceDay** 변수의 값이 1주차를 정의하기 위한 기준일로 설정할 1월 날짜를 정의하는 데 사용됩니다. 기본적으로 Qlik Sense 함수는 4를 기준일로 사용합니다. 이는 1주차에 1월 4일이 포함되어야 한다는 의미입니다. 혹은 달리 말하면 1주차에 항상 1월이 4일 이상이 있어야 합니다.

week() 함수는 날짜가 속하는 주를 확인하고 주차를 반환합니다.

Qlik Sense에서는 앱이 만들어질 때 지역 설정을 가져오고 해당 설정은 스크립트에 환경 변수로 저장됩니다. 이들은 주차를 확인하는 데 사용됩니다.

따라서 대부분의 유럽 앱 개발자는 ISO 8601 정의에 해당하는 다음과 같은 환경 변수를 얻습니다.

```
Set FirstWeekDay =0;    // Monday as first week day
Set BrokenWeeks  =0;    // Use unbroken weeks
Set ReferenceDay =4;    // Jan 4th is always in week 1
```

북미 앱 개발자는 종종 다음과 같은 환경 변수를 얻습니다.

```
Set FirstWeekDay =6;    // Sunday as first week day
Set BrokenWeeks  =1;    // Use broken weeks
Set ReferenceDay =1;    // Jan 1st is always in week 1
```

주의 첫 번째 요일은 FirstWeekDay 시스템 변수에 의해 결정됩니다. week() 함수의 first_week_day 인수를 사용하여 주의 첫 번째 요일을 변경할 수도 있습니다.

응용 프로그램이 분리된 주를 사용하는 경우 주차 계산은 발생한 일 수에 관계없이 1월 1일에 시작하여 FirstWeekDay 시스템 변수 이전 날에 끝납니다.

응용 프로그램이 분리되지 않은 주를 사용하는 경우 1주는 전년도 또는 1월의 처음 며칠에 시작할 수 있습니다. 이는 FirstWeekDay 및 ReferenceDay 환경 변수의 사용 방법에 따라 다릅니다.

사용 시기

The week() 함수는 주별로 집계를 비교하려는 경우에 유용합니다. 예를 들어, 주별 제품의 총 판매량을 보고 싶을 때 사용할 수 있습니다. 사용자가 응용 프로그램의 BrokenWeeks, FirstWeekDay 또는 ReferenceDay 시스템 변수를 사용하지 않아도 되는 계산을 원할 때 weekname()보다 week() 함수를 선택합니다.

예를 들어 주별 제품의 총 판매액을 보려는 경우.

응용 프로그램이 분리되지 않은 주를 사용하는 경우 1주차는 전년도 12월의 날짜를 포함하거나 현재 연도의 1월 날짜를 제외할 수 있습니다. 응용 프로그램에서 분리된 주를 사용하는 경우 1주차에는 7일 미만이 포함될 수 있습니다.

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

아래의 예는 다음과 같이 가정합니다.

```
Set DateFormat= 'MM/DD/YYYY';
Set FirstWeekDay=0;
Set BrokenWeeks=0;
Set ReferenceDay=4;
```

함수 예

예	결과
week('12/28/2021')	52를 반환합니다.
week(44614)	8을 반환합니다(02/22/2022에 대한 일련 번호임).
week('01/03/2021')	53을 반환합니다.
week('01/03/2021',6)	1을 반환합니다.

예 1 - 기본 시스템 변수

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 2021년의 마지막 주와 2022년의 처음 2주 동안의 트랜잭션 집합이 포함된 데이터 집합이며, Transactions라는 테이블에 로드됩니다.
- DateFormat 시스템 변수(MM/DD/YYYY) 서식으로 제공된 날짜 필드.
- 트랜잭션이 발생한 연도 및 주차를 반환하는 필드 week_number 만들기.
- 각 트랜잭션 날짜의 요일 값을 표시하는 week_day라는 필드 만들기.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- id
- date
- week_day
- week_number

결과 테이블

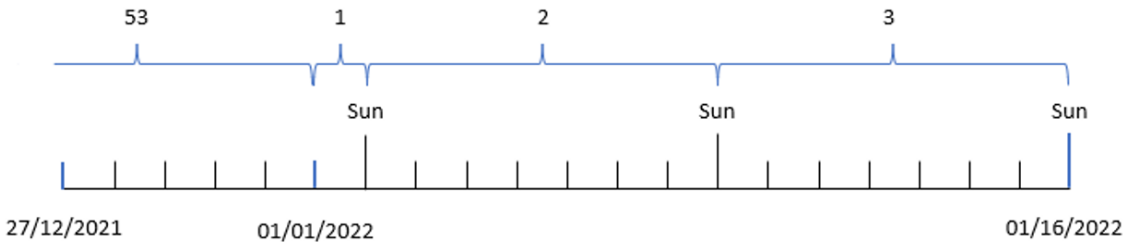
id	date	week_day	week_number
8183	12/27/2021	Mon	53
8184	12/28/2021	Tue	53
8185	12/29/2021	Wed	53
8186	12/30/2021	Thu	53
8187	12/31/2021	Fri	53
8188	01/01/2022	Sat	1
8189	01/02/2022	Sun	2
8190	01/03/2022	Mon	2
8191	01/04/2022	Tue	2
8192	01/05/2022	Wed	2
8193	01/06/2022	Thu	2
8194	01/07/2022	Fri	2
8195	01/08/2022	Sat	2
8196	01/09/2022	Sun	3
8197	01/10/2022	Mon	3
8198	01/11/2022	Tue	3
8199	01/12/2022	Wed	3
8200	01/13/2022	Thu	3
8201	01/14/2022	Fri	3

week_number 필드는 week() 함수를 사용하고 date 필드를 함수의 인수로 전달하여 선행 LOAD 문에서 만들어집니다.

다른 매개 변수는 함수에 전달되지 않으므로 week() 함수에 영향을 미치는 다음 기본 변수가 적용됩니다.

- BrokenWeeks: 주 계산은 1월 1일에 시작됨
- FirstWeekDay: 한 주의 첫날은 일요일

기본 시스템 변수를 사용하는 week() 함수의 다이어그램



응용 프로그램이 기본 BrokenWeeks 시스템 변수를 사용하기 때문에 1주차는 토요일인 1월 1일에 시작됩니다.

기본 FirstWeekDay 시스템 변수로 인해 주는 일요일에 시작됩니다. 1월 1일 이후의 첫 번째 일요일은 2주차가 시작되는 1월 2일에 발생합니다.

예 2 – first_week_day

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 트랜잭션이 발생한 연도 및 주차를 반환하는 필드 week_number 만들기.
- 각 트랜잭션 날짜의 요일 값을 표시하는 week_day라는 필드 만들기.

이 예에서는 근무 주의 시작을 화요일로 설정하려고 합니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- id
- date
- week_day
- week_number

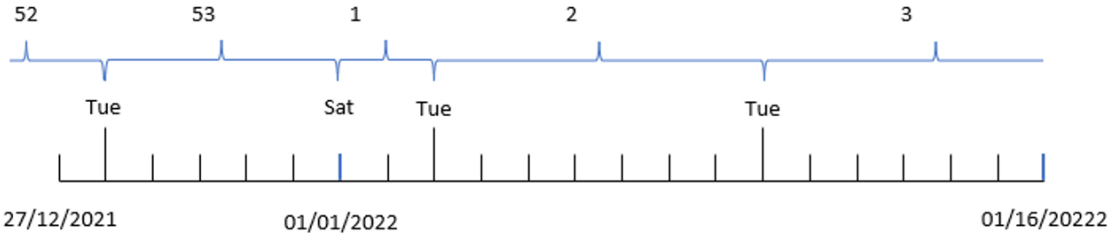
결과 테이블

id	date	week_day	week_number
8183	12/27/2021	Mon	52
8184	12/28/2021	Tue	53
8185	12/29/2021	Wed	53
8186	12/30/2021	Thu	53
8187	12/31/2021	Fri	53
8188	01/01/2022	Sat	1
8189	01/02/2022	Sun	1
8190	01/03/2022	Mon	1
8191	01/04/2022	Tue	2

id	date	week_day	week_number
8192	01/05/2022	Wed	2
8193	01/06/2022	Thu	2
8194	01/07/2022	Fri	2
8195	01/08/2022	Sat	2
8196	01/09/2022	Sun	2
8197	01/10/2022	Mon	2
8198	01/11/2022	Tue	3
8199	01/12/2022	Wed	3
8200	01/13/2022	Thu	3
8201	01/14/2022	Fri	3

응용 프로그램은 여전히 분리된 주를 사용하고 있습니다. 그러나 `first_week_day` 인수는 `week()` 함수에서 1로 설정되었습니다. 이렇게 하면 주의 첫 번째 요일이 화요일로 설정됩니다.

week() 함수 다이어그램, first_week_day 예



응용 프로그램은 기본 `BrokenWeeks` 시스템 변수를 사용하므로 1주차는 토요일인 1월 1일에 시작됩니다.

`week()` 함수의 `first_week_day` 인수는 첫 번째 요일을 화요일로 설정합니다. 따라서 53주차는 2021년 12월 28일에 시작됩니다.

그러나 이 함수가 여전히 분리된 주를 사용하므로 1월 1일 이후의 첫 번째 화요일이 1월 3일이기 때문에 1주차는 2일만 됩니다.

예 3 – unbroken_weeks

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

이 로드 스크립트에는 첫 번째 예와 동일한 데이터 집합 및 시나리오가 포함되어 있습니다.

이 예에서는 분리되지 않은 주를 사용합니다.

로드 스크립트

```
SET DateFormat='MM/DD/YYYY';
SET FirstWeekDay=6;
SET BrokenWeeks=1;
SET ReferenceDay=0;

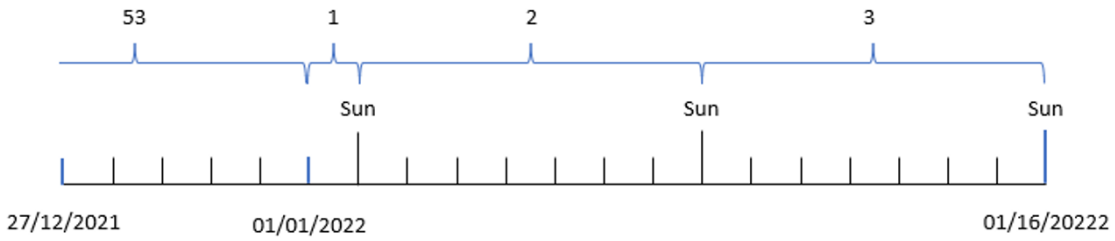
Transactions:
    Load
        *,
        WeekDay(date) as week_day,
        Week(date,6,0) as week_number
        ;
Load
*
Inline
[
id,date,amount
8183,12/27/2022,58.27
8184,12/28/2022,67.42
8185,12/29/2022,23.80
8186,12/30/2022,82.06
8187,12/31/2021,40.56
8188,01/01/2022,37.23
8189,01/02/2022,17.17
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- id
- date
- week_day
- week_number

week() 함수의 다이어그램, 차트 개체 예



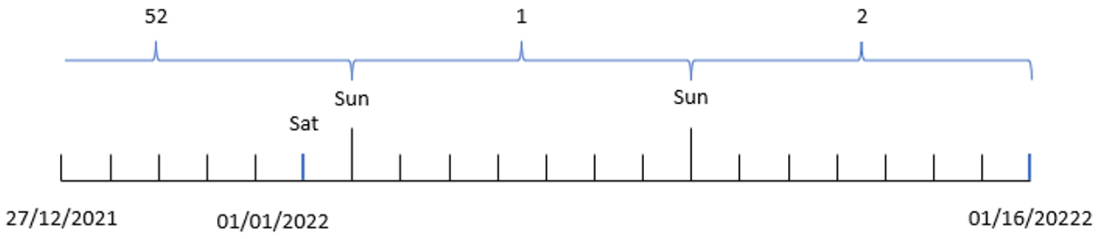
결과 테이블

id	date	week_day	week_number
8183	12/27/2021	Mon	52
8184	12/28/2021	Tue	52
8185	12/29/2021	Wed	52
8186	12/30/2021	Thu	52
8187	12/31/2021	Fri	52
8188	01/01/2022	Sat	52
8189	01/02/2022	Sun	1
8190	01/03/2022	Mon	1
8191	01/04/2022	Tue	1
8192	01/05/2022	Wed	1
8193	01/06/2022	Thu	1
8194	01/07/2022	Fri	1
8195	01/08/2022	Sat	1
8196	01/09/2022	Sun	2
8197	01/10/2022	Mon	2
8198	01/11/2022	Tue	2
8199	01/12/2022	Wed	2
8200	01/13/2022	Thu	2
8201	01/14/2022	Fri	2

first_week_date 매개 변수는 1로 설정되어 화요일을 주의 첫 번째 요일로 만듭니다. broken_weeks 매개 변수는 0으로 설정하여 이 함수가 분리되지 않은 주를 사용하도록 합니다. 마지막으로 세 번째 매개 변수는 reference_day를 2로 설정합니다.

first_week_date 매개 변수는 6으로 설정되어 일요일을 주의 첫 번째 요일로 만듭니다. broken_weeks 매개 변수는 0으로 설정되어 이 함수가 분리되지 않은 주를 사용하도록 합니다.

week() 함수의 다이어그램, 분리되지 않은 주를 사용한 예



분리되지 않은 주를 사용하면 1주차가 반드시 1월 1일에 시작되는 것은 아닙니다. 대신 최소 4일이 필요합니다. 따라서 데이터 집합에서 52주차는 2022년 1월 1일 토요일에 끝납니다. 그런 다음 1주차는 FirstWeekDay 시스템 변수에서 시작되며 이는 1월 2일 일요일입니다. 이번 주는 다음 1월 8일 토요일에 종료됩니다.

예 4 – reference_day

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 세 번째 예와 동일한 데이터 집합 및 시나리오.
- 트랜잭션이 발생한 연도 및 주차를 반환하는 필드 week_number 만들기.
- 각 트랜잭션 날짜의 요일 값을 표시하는 week_day라는 필드 만들기.

또한 다음 조건을 충족해야 합니다.

- 근무 주가 화요일에 시작됩니다.
- 회사는 분리되지 않은 주를 사용합니다.
- reference_day 값은 2입니다. 즉, 1주차의 1월 최소 일수는 2입니다.

로드 스크립트

```
SET DateFormat='MM/DD/YYYY';
SET FirstWeekDay=6;
SET BrokenWeeks=1;
SET ReferenceDay=0;

Transcactions:
    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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- id
- date
- week_day
- week_number

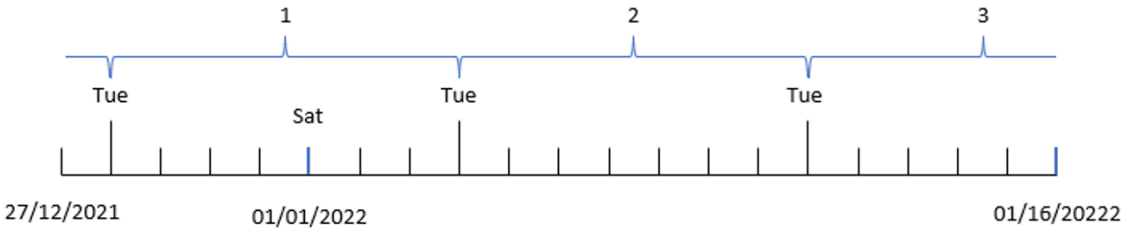
결과 테이블

id	date	week_day	week_number
8183	12/27/2021	Mon	52
8184	12/28/2021	Tue	1
8185	12/29/2021	Wed	1
8186	12/30/2021	Thu	1
8187	12/31/2021	Fri	1
8188	01/01/2022	Sat	1
8189	01/02/2022	Sun	1

id	date	week_day	week_number
8190	01/03/2022	Mon	1
8191	01/04/2022	Tue	2
8192	01/05/2022	Wed	2
8193	01/06/2022	Thu	2
8194	01/07/2022	Fri	2
8195	01/08/2022	Sat	2
8196	01/09/2022	Sun	2
8197	01/10/2022	Mon	2
8198	01/11/2022	Tue	3
8199	01/12/2022	Wed	3
8200	01/13/2022	Thu	3
8201	01/14/2022	Fri	3

first_week_date 매개 변수는 1로 설정되어 화요일을 주의 첫 번째 요일로 만듭니다. broken_weeks 매개 변수는 0으로 설정되어 이 함수가 분리되지 않은 주를 사용하도록 합니다. 마지막으로 세 번째 매개 변수는 reference_day 매개 변수를 2로 설정합니다.

week() 함수 다이어그램, reference_day 예



이 함수에서 분리되지 않은 주를 사용하고 reference_day 값 2를 매개 변수로 사용하는 경우 1주차는 1월의 2일만 포함하면 됩니다. 첫 번째 날이 화요일이므로 1주차는 2021년 12월 28일에 시작하여 2022년 1월 3일 월요일에 끝납니다.

예 5 – 차트 개체 예

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

이 로드 스크립트에는 첫 번째 예와 동일한 데이터 집합 및 시나리오가 포함되어 있습니다.

그러나 이 예에서는 변경되지 않은 데이터 집합이 응용 프로그램에 로드됩니다. 주차를 반환하는 계산은 차트 개체에서 측정값으로 만들어집니다.

로드 스크립트

```
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
];
```

결과

다음과 같이 하십시오.

1. 데이터를 로드하고 시트를 엽니다. 새 테이블을 만듭니다.
2. 다음 필드를 차원으로 추가합니다.
   - id
   - date
3. 다음으로 다음 측정값을 만듭니다.
   `=week (date)`
4. 각 트랜잭션 날짜의 요일 값을 표시하는 측정값, week_day를 만듭니다.
   `=weekday(date)`

결과 테이블

id	date	=week(date)	=weekday(date)
8183	12/27/2021	53	Mon
8184	12/28/2021	53	Tue

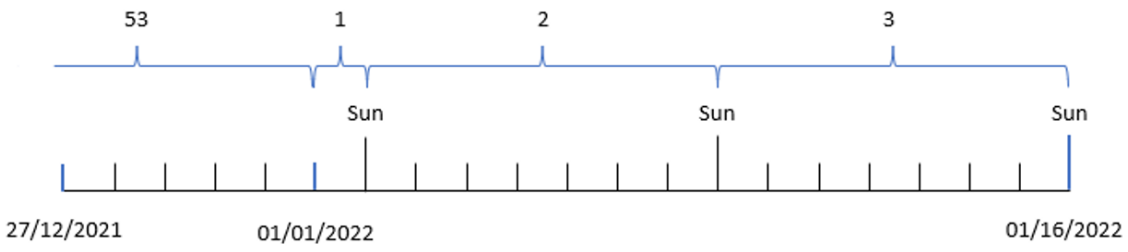
id	date	=week(date)	=weekday(date)
8185	12/29/2021	53	Wed
8186	12/30/2021	53	Thu
8187	12/31/2021	53	Fri
8188	01/01/2022	1	Sat
8189	01/02/2022	2	Sun
8190	01/03/2022	2	Mon
8191	01/04/2022	2	Tue
8192	01/05/2022	2	Wed
8193	01/06/2022	2	Thu
8194	01/07/2022	2	Fri
8195	01/08/2022	2	Sat
8196	01/09/2022	3	Sun
8197	01/10/2022	3	Mon
8198	01/11/2022	3	Tue
8199	01/12/2022	3	Wed
8200	01/13/2022	3	Thu
8201	01/14/2022	3	Fri

week_number 필드는 week() 함수를 사용하고 date 필드를 함수의 인수로 전달하여 선행 LOAD 문에서 만들어집니다.

다른 매개 변수는 함수에 전달되지 않으므로 week() 함수에 영향을 미치는 다음 기본 변수가 적용됩니다.

- BrokenWeeks: 주 계산은 1월 1일에 시작됨
- FirstWeekDay: 한 주의 첫날은 일요일

week() 함수의 다이어그램, 차트 개체 예



응용 프로그램이 기본 BrokenWeeks 시스템 변수를 사용하기 때문에 1주차는 토요일인 1월 1일에 시작됩니다.

기본 FirstWeekDay 시스템 변수로 인해 주는 일요일에 시작됩니다. 1월 1일 이후의 첫 번째 일요일은 2주차가 시작되는 1월 2일에 발생합니다.

예 6 – 시나리오

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Transactions라는 테이블에 로드되는 2019년의 마지막 주와 2020년의 처음 2주 동안의 트랜잭션 집합을 포함하는 데이터 집합.
- DateFormat 시스템 변수(MM/DD/YYYY) 서식으로 제공된 날짜 필드.

응용 프로그램은 주로 대시보드에서 분리된 주를 사용합니다. 그러나 최종 사용자는 분리되지 않은 주를 사용하여 주별 총 판매를 표시하는 차트 개체를 원합니다. 기준일은 1월 2일이어야 하며 주는 화요일에 시작해야 합니다. 데이터 모델에서 이 차원을 사용할 수 없는 경우에도 week() 함수를 차트에서 계산된 차원으로 사용하여 이를 달성할 수 있습니다.

로드 스크립트

```
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
];
```

결과

다음과 같이 하십시오.

1. 데이터를 로드하고 시트를 엽니다. 새 테이블을 만듭니다.
2. 다음 계산된 차원을 만듭니다.
   `=week(date)`
3. 다음으로, 다음 집계 측정값을 만듭니다.
   `=sum(amount)`
4. 측정값의 **숫자 형식**을 **화폐**로 설정합니다.
5. **정렬** 메뉴를 선택하고 계산된 차원에 대해 사용자 지정 정렬을 제거합니다.
6. **숫자순 정렬** 및 **사전순 정렬** 옵션을 선택 취소합니다.

결과 테이블

week(date)	sum(amount)
52	$125.69
53	$146.42
1	$200.09
2	$347.57
3	$122.01

weekday

이 함수는 다음을 포함하는 이중 값을 반환합니다.

- 환경 변수 **DayNames**로 정의한 날짜 이름.
- 주의 명목상 이름(0~6)에 해당하는 0~6 사이의 정수.

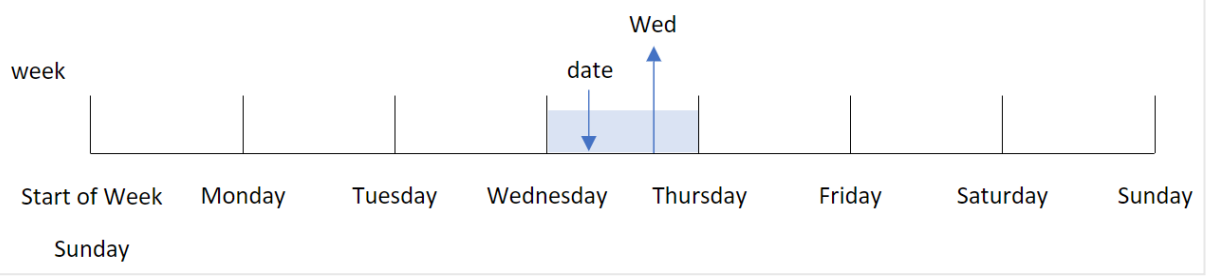
구문:

weekday(date [,**first_week_day**=0]**)**

반환 데이터 유형: dual

weekday() 함수는 날짜가 발생하는 요일을 결정합니다. 그런 다음 해당 날짜를 나타내는 문자열 값을 반환합니다.

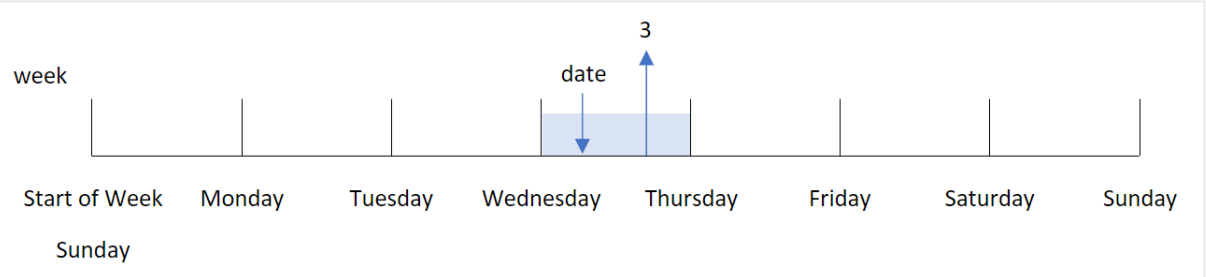
날짜가 속하는 날짜의 이름을 반환하는 weekday() 함수의 다이어그램



결과는 주의 시작일을 기준으로 해당 요일에 해당하는 숫자 값(0-6)을 반환합니다. 예를 들어, 주의 첫 번째 요일이 일요일로 설정된 경우 수요일은 숫자 값 3을 반환합니다. 이 시작일은 FirstWeekDay 시스템 변수나 first_week_day 함수 매개 변수에 의해 결정됩니다.

이 숫자 값을 산술 표현식의 일부로 사용할 수 있습니다. 예를 들어, 1로 곱하면 값 자체가 반환됩니다.

요일 이름 대신 요일의 숫자 값이 표시되는 weekday() 함수의 다이어그램



사용 시기

weekday() 함수는 요일별로 집계를 비교하려는 경우에 유용합니다. 예를 들어 요일별 평균 제품 판매를 비교하려는 경우에 사용할 수 있습니다.

이러한 차원은 로드 스크립트에서 **마스터 캘린더** 테이블에 필드를 만드는 함수를 사용하여 만들 수 있습니다. 또는 차트에서 계산된 측정값으로 직접 만들어집니다.

관련 항목

항목	상호 작용
FirstWeekDay (page 219)	각 주의 시작 날짜를 정의합니다.

인수

인수	설명
date	평가할 날짜 또는 타임스탬프입니다.
first_week_day	주의 시작 요일을 지정합니다. 생략하면 변수 **FirstWeekDay**의 값이 사용됩니다. *FirstWeekDay (page 219)*

다음 값을 사용하여 first_week_day 인수에서 주가 시작되는 날짜를 설정할 수 있습니다.

first_week_day 값

일	Value
월요일	0
화요일	1
수요일	2
목요일	3
금요일	4
토요일	5
일요일	6

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

언급하지 않는 한, 이 예에서 FirstWeekDay는 0으로 설정됩니다.

함수 예

예	결과
weekday('10/12/1971')	'Tue' 및 1을 반환합니다.
weekday('10/12/1971' , 6)	'Tue' 및 2를 반환합니다. 이 예에서는 일요일(6)이 주의 시작 날짜입니다.
SET FirstWeekDay=6; ... weekday('10/12/1971')	'Tue' 및 2를 반환합니다.

예 1 - 요일 문자열

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 'Transactions'라는 테이블에 로드되는 2022년 트랜잭션 집합이 포함된 데이터 집합.
- 6(일요일)으로 설정된 FirstWeekDay 시스템 변수.
- 기본 요일 이름을 사용하도록 설정된 DayNames 변수.
- 'week_day' 필드로 설정되고 트랜잭션이 발생한 요일을 반환하는 weekday() 함수가 포함된 선행 LOAD.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- id
- date
- week_day

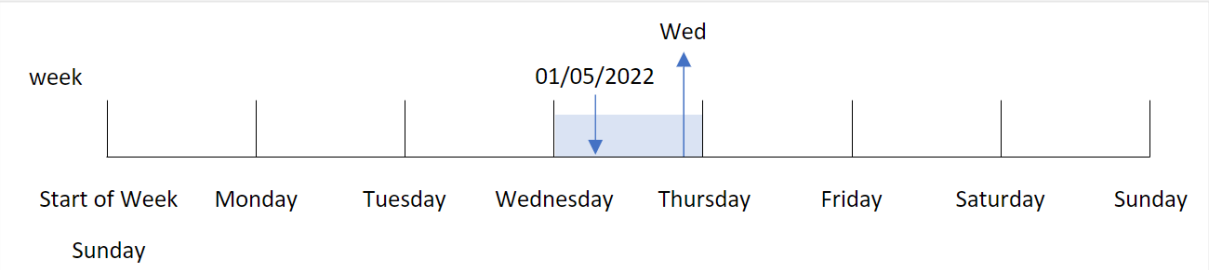
결과 테이블

id	date	week_day
8188	01/01/2022	Sat
8189	01/02/2022	Sun
8190	01/03/2022	Mon
8191	01/04/2022	Tue
8192	01/05/2022	Wed
8193	01/06/2022	Thu
8194	01/07/2022	Fri

'week_day' 필드는 weekday() 함수를 사용하고 날짜 필드를 함수의 인수로 전달하여 선행 LOAD 문에서 만들어집니다.

weekday() 함수는 요일 문자열 값을 반환합니다. 즉, DayNames 시스템 변수에 의해 설정된 요일의 이름을 반환합니다.

수요일을 트랜잭션 8192의 요일로 반환하는 weekday() 함수의 다이어그램



트랜잭션 8192는 1월 5일에 발생했습니다. FirstWeekDay 시스템 변수는 주의 첫 번째 요일을 일요일로 설정합니다. weekday() 함수 트랜잭션은 수요일에 발생했으며 이 값을 DayNames 시스템 변수의 축약 형식으로 week_day 필드에 반환합니다.

필드(수요일, 3)에 대한 이중 숫자 및 텍스트 결과가 있기 때문에 'week_day' 필드의 값은 열에서 오른쪽 정렬됩니다. 필드 값을 해당하는 숫자로 변환하기 위해 필드를 num() 함수 내에서 래핑할 수 있습니다. 예를 들어 트랜잭션 8192에서 수요일 값은 숫자 3으로 변환됩니다.

예 2 – first_week_day

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 'Transactions'라는 테이블에 로드되는 2022년 트랜잭션 집합이 포함된 데이터 집합.
- 6(일요일)으로 설정된 FirstWeekDay 시스템 변수.
- 기본 요일 이름을 사용하도록 설정된 DayNames 변수.
- 'week_day' 필드로 설정되고 트랜잭션이 발생한 요일을 반환하는 weekday() 함수가 포함된 선행 LOAD.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

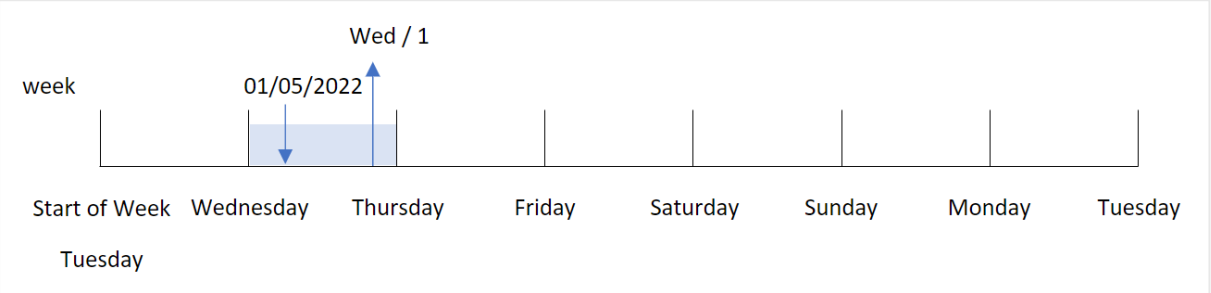
- id
- date
- week_day

결과 테이블

id	date	week_day
8188	01/01/2022	Sat
8189	01/02/2022	Sun
8190	01/03/2022	Mon
8191	01/04/2022	Tue
8192	01/05/2022	Wed

id	date	week_day
8193	01/06/2022	Thu
8194	01/07/2022	Fri

수요일에 이중 숫자 값 1이 있는 것을 보여 주는 weekday() 함수의 다이어그램



first_week_day 인수가 weekday() 함수에서 1로 설정되어 있으므로 주의 첫 번째 요일은 화요일입니다. 따라서 화요일에 발생하는 모든 트랜잭션에는 이중 숫자 값 0이 있습니다.

트랜잭션 8192는 1월 5일에 발생했습니다. weekday() 함수는 이것이 수요일임을 식별하므로 표현식은 이중 숫자 값 1을 반환합니다.

예 3 - 차트 개체 예

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 'Transactions'라는 테이블에 로드되는 2022년 트랜잭션 집합이 포함된 데이터 집합.
- 6(일요일)으로 설정된 FirstWeekDay 시스템 변수.
- 기본 요일 이름을 사용하도록 설정된 DayNames 변수.

그러나 이 예에서 데이터 집합은 변경되지 않고 응용 프로그램에 로드됩니다. 요일 값을 식별하는 계산은 앱의 차트에서 측정값으로 만들어집니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- id
- date

요일 값을 계산하려면 다음 측정값을 만듭니다.

- =weekday(date)

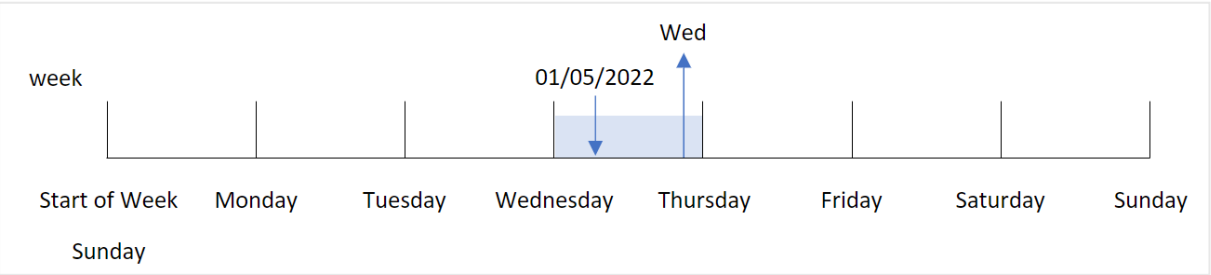
결과 테이블

id	date	=weekday(date)
8188	01/01/2022	Sat
8189	01/02/2022	Sun
8190	01/03/2022	Mon
8191	01/04/2022	Tue
8192	01/05/2022	Wed
8193	01/06/2022	Thu
8194	01/07/2022	Fri

'=weekday(date)' 필드는 weekday() 함수를 사용하고 함수의 인수로 날짜 필드를 전달하여 차트에서 만들어집니다.

weekday() 함수는 요일 문자열 값을 반환합니다. 즉, DayNames 시스템 변수에 의해 설정된 요일의 이름을 반환합니다.

수요일을 트랜잭션 8192의 요일로 반환하는 weekday() 함수의 다이어그램



트랜잭션 8192는 1월 5일에 발생했습니다. FirstWeekDay 시스템 변수는 주의 첫 번째 요일을 일요일로 설정합니다. weekday() 함수 트랜잭션은 수요일에 발생했으며 이 값을 DayNames 시스템 변수의 축약 형식으로 =weekday(date) 필드에 반환합니다.

예 4 - 시나리오

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 'Transactions'라는 테이블에 로드되는 2022년 트랜잭션 집합이 포함된 데이터 집합.
- 6(일요일)으로 설정된 FirstWeekDay 시스템 변수.
- 기본 요일 이름을 사용하도록 설정된 DayNames 변수.

최종 사용자는 트랜잭션에 대한 요일별 평균 판매를 나타내는 차트를 원합니다.

로드 스크립트

```
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);
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- =weekday(date)
- =avg(amount)

측정값의 **숫자 형식**을 **화폐**로 설정합니다.

결과 테이블

weekday(date)	Avg(amount)
Sun	$536.96
Mon	$500.80
Tue	$515.63

weekday(date)	Avg(amount)
Wed	$509.21
Thu	$482.70
Fri	$441.33
Sat	$505.22

weekend

이 함수는 **date**를 포함하는 캘린더 주의 마지막 날 마지막 밀리초의 타임스탬프에 해당하는 값을 반환합니다. 기본 출력 형식은 스크립트에 설정된 **DateFormat**입니다.

구문:

```
WeekEnd(timestamp [, period_no [, first_week_day ]])
```

반환 데이터 유형: dual

weekend() 함수는 날짜가 속하는 주를 결정합니다. 그런 다음 해당 주의 마지막 밀리초에 대한 타임스탬프를 날짜 서식으로 반환합니다. 주의 첫 번째 요일은 FirstWeekDay 환경 변수에 의해 결정됩니다. 그러나 이는 weekend() 함수의 first_week_day 인수로 대체될 수 있습니다.

인수

인수	설명
timestamp	평가할 날짜 또는 타임스탬프입니다.
period_no	**shift**는 정수이며, 값 0은 **date**를 포함하는 주를 나타냅니다. shift가 음수 값일 경우 이전 주, 양수 값일 경우 다음 주를 나타냅니다.
first_week_day	주의 시작 요일을 지정합니다. 생략하면 변수 **FirstWeekDay**의 값이 사용됩니다. **first_week_day**의 가능한 값은 월요일에 0, 화요일에 1, 수요일에 2, 목요일에 3, 금요일에 4, 토요일에 5, 일요일에 6입니다. 시스템 변수에 대한 자세한 내용은 *FirstWeekDay (page 219)*를 참조하십시오.

사용 시기

weekend() 함수는 일반적으로 사용자가 지정된 날짜에 대해 해당 주의 나머지 일 수를 계산하려고 할 때 표현식의 일부로 사용됩니다. 예를 들어, 사용자가 한 주 동안 아직 발생하지 않은 총 이자를 계산하려는 경우 사용할 수 있습니다.

다음 예에서는 다음을 가정합니다.

```
SET FirstWeekDay=0;
```

예	결과
weekend('01/10/2013')	01/12/2013 23:59:59를 반환합니다.
weekend('01/10/2013', -1)	01/05/2013 23:59:59.를 반환합니다.
weekend('01/10/2013', 0, 1)	01/14/2013 23:59:59를 반환합니다.

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

주 및 주차에 대해 ISO 설정을 사용하려는 경우 스크립트에 다음이 포함되어 있는지 확인합니다.

```
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
```

미국 설정을 사용하려는 경우 스크립트에 다음이 포함되어 있는지 확인합니다.

```
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
```

위의 예는 weekend() 함수에서 다음과 같은 결과를 가져옵니다.

Weekend 함수의 예

Date	ISO 주 끝	미국 주 끝
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

Date	ISO 주 끝	미국 주 끝
Sat 2021 Jan 2	2021-01-03	1/2/2021
Sun 2021 Jan 3	2021-01-03	1/9/2021
Mon 2021 Jan 4	2021-01-10	1/9/2021
Tue 2021 Jan 5	2021-01-10	1/9/2021

주의 끝은 ISO 열에서 일요일이고 미국 열에서 토요일입니다.

예 1 – 기본 예

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Transactions라는 테이블에 로드되는 2022년 트랜잭션 집합이 포함된 데이터 집합.
- DateFormat 시스템 변수(MM/DD/YYYY) 서식으로 제공된 날짜 필드.
- 트랜잭션이 발생한 주의 마지막에 대한 타임스탬프를 반환하는 필드 end_of_week 만들기.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- end_of_week
- end_of_week_timestamp

결과 테이블

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

date	end_of_week	end_of_week_timestamp
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

end_of_week 필드는 weekend() 함수를 사용하고 날짜 필드를 함수의 인수로 전달하여 선행 LOAD 문에서 만들어집니다.

weekend() 함수는 날짜 값이 속하는 주를 식별하고 해당 주의 마지막 밀리초에 대한 타임스탬프를 반환합니다.

weekend() 함수의 다이어그램, 기본 예

트랜잭션 8191은 2월 5일에 발생했습니다. FirstWeekDay 시스템 변수는 주의 첫 번째 요일을 일요일로 설정합니다. weekend() 함수는 2월 5일 이후의 첫 번째 토요일(따라서 주의 끝)이 2월 5일임을 식별합니다. 따라서 해당 트랜잭션의 end_of_week 값은 해당 날짜의 마지막 밀리초인 2월 5일 오후 11:59:59를 반환합니다.

예 2 – period_no

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 첫 번째 예와 동일한 데이터 집합 및 시나리오.
- 트랜잭션이 발생하기 전 주의 시작에 대한 타임스탬프를 반환하는 필드 previous_week_end 만들기.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- previous_week_end
- previous_week_end_timestamp

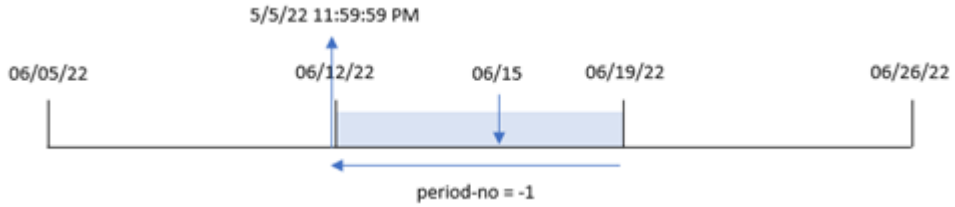
결과 테이블

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

date	end_of_week	end_of_week_timestamp
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

이 경우 weekend() 함수에서 period_no -1을 오프셋 인수로 사용했으므로 이 함수는 먼저 트랜잭션이 발생한 주를 식별합니다. 그런 다음 1주일 전을 찾아 해당 주의 마지막 밀리초를 식별합니다.

weekend() 함수의 다이어그램, period_no 예



트랜잭션 8196은 6월 15일에 발생했습니다. weekend() 함수는 주가 6월 12일에 시작함을 식별합니다. 따라서 이전 주는 6월 11일 오후 11:59:59에 끝납니다. 이는 previous_week_end 필드에 대해 반환된 값입니다.

예 3 – first_week_day

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

이 로드 스크립트에는 첫 번째 예와 동일한 데이터 집합 및 시나리오가 포함되어 있습니다. 그러나 이 예에서는 화요일을 근무 주의 첫 번째 요일로 설정해야 합니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- end_of_week
- end_of_week_timestamp

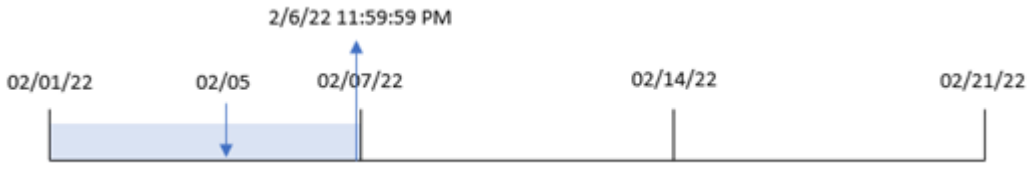
결과 테이블

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

date	end_of_week	end_of_week_timestamp
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

이 경우 weekend() 함수에서 first_week_date 인수 1을 사용하므로 주의 첫 번째 날을 화요일로 설정합니다.

weekend() 함수 다이어그램, first_week_day 예



트랜잭션 8191은 2월 5일에 발생했습니다. weekend() 함수는 이 날짜 이후의 첫 번째 월요일(따라서 한 주의 끝과 반환된 값)이 2월 6일 오후 11:59:59임을 식별합니다.

예 4 – 차트 개체 예

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

이 로드 스크립트에는 첫 번째 예와 동일한 데이터 집합 및 시나리오가 포함되어 있습니다. 그러나 이 예에서는 변경되지 않은 데이터 집합이 응용 프로그램에 로드됩니다. 트랜잭션이 발생한 주의 끝에 대한 타임스탬프를 반환하는 계산은 응용 프로그램의 차트 개체에서 측정값으로 만들어집니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다. `date`.

트랜잭션이 발생한 주의 시작을 계산하려면 다음 측정값을 추가합니다.

- `=weekend(date)`
- `=timestamp(weekend(date))`

결과 테이블

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

date	=weekend(date)	=timestamp(weekend(date))
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

end_of_week 측정값은 weekend() 함수를 사용하고 날짜 필드를 함수의 인수로 전달하여 차트 개체에서 만들어집니다. weekend() 함수는 날짜 값이 속하는 주를 식별하고 해당 주의 마지막 밀리초에 대한 타임스탬프를 반환합니다.

weekend() 함수의 다이어그램, 차트 개체 예

트랜잭션 8191은 2월 5일에 발생했습니다. FirstWeekDay 시스템 변수는 주의 첫 번째 요일을 일요일로 설정합니다. weekend() 함수는 2월 5일 이후의 첫 번째 토요일(따라서 주의 끝)이 2월 5일임을 식별합니다. 따라서 해당 트랜잭션의 end_of_week 값은 해당 날짜의 마지막 밀리초인 2월 5일 오후 11:59:59를 반환합니다.

예 5 – 시나리오

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Employee_Expenses라는 테이블에 로드되는 데이터 집합.
- 직원 ID, 직원 이름 및 각 직원의 평균 일일 비용 클레임으로 구성된 데이터.

최종 사용자는 직원 ID와 직원 이름별로 해당 주에 남은 기간 동안 발생할 것으로 예상되는 비용 청구를 표시하는 차트 개체를 원합니다.

로드 스크립트

```
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
];
```

결과

다음과 같이 하십시오.

1. 데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.
   - employee_id
   - employee_name
2. 다음으로 누적 이자를 계산하는 측정값을 만듭니다.
   `=(weekend(today(1))-today(1))*avg_daily_claim`
3. 측정값의 **숫자 형식**을 **화폐**로 설정합니다.

결과 테이블

employee_id	employee_name	=(weekend(today(1))-today(1))*avg_daily_claim
182	Mark	$90.00
183	Deryck	$75.00

employee_id	employee_name	=(weekend(today(1))-today(1))*avg_daily_claim
184	Dexter	$75.00
185	Sydney	$162.00
186	Agatha	$108.00

weekend() 함수는 오늘 날짜를 유일한 인수로 사용하여 현재 주의 종료 날짜를 반환합니다. 그런 다음 이 표현식은 주 끝 날짜에서 오늘 날짜를 빼서 이번 주에 남아 있는 일 수를 반환합니다.

그런 다음 이 값에 각 직원의 평균 일일 비용 청구를 곱하여 각 직원이 남은 주에 예상되는 청구 금액을 계산합니다.

weekname

이 함수는 **date**를 포함한 주의 첫 번째 날의 첫 번째 밀리초의 타임스탬프에 해당하는 기본 숫자 값으로 연도 및 주차를 보여주는 값을 반환합니다.

구문:

```
WeekName(date[, period_no [, first_week_day [, broken_weeks [, reference_day]]]])
```

weekname() 함수는 날짜가 속하는 주를 결정하고 주 수와 해당 주의 연도를 반환합니다. 주의 첫 번째 요일은 FirstWeekDay 시스템 변수에 의해 결정됩니다. 그러나 weekname() 함수의 first_week_day 인수를 사용하여 주의 첫 번째 요일을 변경할 수도 있습니다.

Qlik Sense에서는 앱이 만들어질 때 지역 설정을 가져오고 해당 설정은 스크립트에 환경 변수로 저장됩니다.

북미 앱 개발자는 스크립트에서 종종 분리된 주에 해당하는 Set BrokenWeeks=1;을 가져옵니다. 유럽 앱 개발자는 스크립트에서 종종 분리되지 않은 주에 해당하는 Set BrokenWeeks=0;을 가져옵니다.

응용 프로그램이 분리된 주를 사용하는 경우 주차 계산은 발생한 일 수에 관계없이 1월 1일에 시작하여 FirstWeekDay 시스템 변수 전날에 끝납니다.

그러나 응용 프로그램이 분리되지 않은 주를 사용하는 경우 1주차는 전년도 또는 1월의 처음 며칠에 시작할 수 있습니다. 이는 ReferenceDay 및 FirstWeekDay 시스템 변수를 사용하는 방법에 따라 다릅니다.

Weekname 함수의 예

Date	ISO 주 이름	US 주 이름
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

Date	ISO 주 이름	US 주 이름
Fri 2021 Jan 1	2020/53	2021/01
Sat 2021 Jan 2	2020/53	2021/01
Sun 2021 Jan 3	2020/53	2021/02
Mon 2021 Jan 4	2021/01	2021/02
Tue 2021 Jan 5	2021/01	2021/02

사용 시기

weekname() 함수는 주별로 집계를 비교하려는 경우에 유용합니다.

예를 들어 주별 제품의 총 판매액을 보려는 경우. 응용 프로그램에서 BrokenWeeks 환경 변수와의 일관성을 유지하려면 lunarweekname() 대신 weekname()을 사용합니다. 응용 프로그램이 분리되지 않은 주를 사용하는 경우 1주차는 전년도 12월의 날짜를 포함하거나 현재 연도의 1월 날짜를 제외할 수 있습니다. 응용 프로그램에서 분리된 주를 사용하는 경우 1주차에는 7일 미만이 포함될 수 있습니다.

반환 데이터 유형: dual

인수

인수	설명
timestamp	평가할 날짜 또는 타임스탬프입니다.
period_no	**shift**는 정수이며, 값 0은 **date**를 포함하는 주를 나타냅니다. shift가 음수 값일 경우 이전 주, 양수 값일 경우 다음 주를 나타냅니다.
first_week_day	주의 시작 요일을 지정합니다. 생략하면 변수 **FirstWeekDay**의 값이 사용됩니다. **first_week_day**에 가능한 값은 월요일에 0, 화요일에 1, 수요일에 2, 목요일에 3, 금요일에 4, 토요일에 5, 일요일에 6입니다. 시스템 변수에 대한 자세한 내용은 *FirstWeekDay (page 219)*를 참조하십시오.
broken_weeks	**broken_weeks**를 지정하지 않으면 **BrokenWeeks** 변수의 값이 주를 분리할지 여부를 정의하는 데 사용됩니다.
reference_day	**reference_day**를 지정하지 않으면 **ReferenceDay** 변수의 값이 1주차를 정의하기 위한 기준일로 설정할 1월 날짜를 정의하는 데 사용됩니다. 기본적으로 Qlik Sense 함수는 4를 기준일로 사용합니다. 이는 1주차에 1월 4일이 포함되어야 한다는 의미입니다. 혹은 달리 말하면 1주차에 항상 1월이 4일 이상이 있어야 합니다.

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

아래의 예는 다음과 같이 가정합니다.

```
Set FirstWeekDay=0;
Set BrokenWeeks=0;
Set ReferenceDay=4;
```

함수 예

예	결과
weekname('01/12/2013')	2013/02를 반환합니다.
weekname('01/12/2013', -1)	2013/01을 반환합니다.
weekname('01/12/2013', 0, 1)	2013/02를 반환합니다.

예 1 – 추가 인수가 없는 날짜

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 'Transactions'라는 테이블에 로드되는 2021년의 마지막 주와 2022년의 처음 2주 동안의 트랜잭션 집합을 포함하는 데이터 집합.
- MM/DD/YYYY 서식으로 설정된 DateFormat 시스템 변수.
- 1으로 설정된 BrokenWeeks 시스템 변수.
- 6로 설정된 FirstWeekDay 시스템 변수.
- 다음을 포함하는 선행 LOAD:
    - 'week_number' 필드로 설정된 weekday() 함수. 트랜잭션이 발생한 연도와 주차를 반환합니다.
    - 'week_day'라는 필드로 설정된 weekname() 함수. 각 트랜잭션 날짜의 요일 값을 표시합니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- id
- date
- week_day
- week_number

결과 테이블

id	date	week_day	week_number
8183	12/27/2021	Mon	2021/53
8184	12/28/2021	Tue	2021/53
8185	12/29/2021	Wed	2021/53
8186	12/30/2021	Thu	2021/53
8187	12/31/2021	Fri	2021/53
8188	01/01/2022	Sat	2022/01
8189	01/02/2022	Sun	2022/02

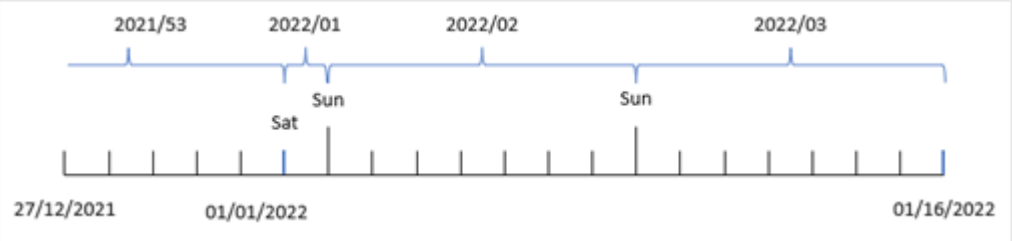
id	date	week_day	week_number
8190	01/03/2022	Mon	2022/02
8191	01/04/2022	Tue	2022/02
8192	01/05/2022	Wed	2022/02
8193	01/06/2022	Thu	2022/02
8194	01/07/2022	Fri	2022/02
8195	01/08/2022	Sat	2022/02
8196	01/09/2022	Sun	2022/03
8197	01/10/2022	Mon	2022/03
8198	01/11/2022	Tue	2022/03
8199	01/12/2022	Wed	2022/03
8200	01/13/2022	Thu	2022/03
8201	01/14/2022	Fri	2022/03

'week_number' 필드는 weekname() 함수를 사용하고 날짜 필드를 함수의 인수로 전달하여 선행 LOAD 문에서 만들어집니다.

weekname() 함수는 처음에 날짜 값이 속하는 주를 식별하고 트랜잭션이 발생한 연도와 주 수 개수를 반환합니다.

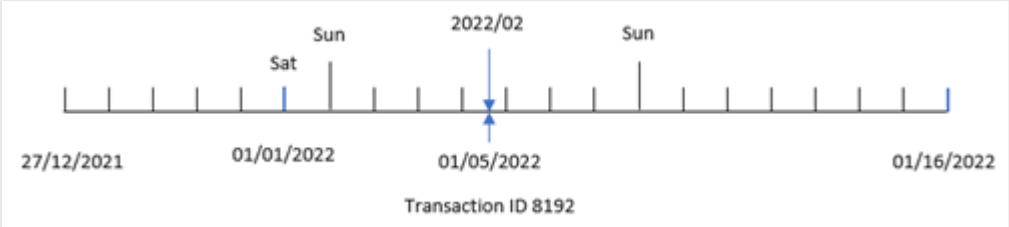
FirstWeekDay 시스템 변수는 일요일을 주의 첫 번째 요일로 설정합니다. BrokenWeeks 시스템 변수는 분리된 주를 사용하도록 응용 프로그램을 설정합니다. 즉, 1주차는 1월 1일에 시작됩니다.

기본 변수가 있는 weekname() 함수 다이어그램.



1주차는 토요일인 1월 1일에 시작하므로 이 날짜에 발생한 트랜잭션은 값 2022/01(연도 및 주 수)을 반환합니다.

트랜잭션 8192의 주 수를 식별하는 weekname() 함수 다이어그램.



응용 프로그램이 분리된 주를 사용하고 첫 번째 요일이 일요일이기 때문에 1월 2일부터 1월 8일까지 발생한 트랜잭션은 값 2022/02(2022년의 주 수 2)를 반환합니다. 이에 대한 예는 1월 5일에 발생한 트랜잭션 8192이며 'week_number' 필드의 값 2022/02를 반환합니다.

예 2 – period_no

로드 스크립트 및 결과

개요

첫 번째 예와 동일한 데이터 집합 및 시나리오가 사용됩니다.

그러나 이 예에서 작업은 트랜잭션이 발생한 이전 연도 및 주 수를 반환하는 필드 'previous_week_number'를 만드는 것입니다.

데이터 로드 편집기를 열고 다음의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

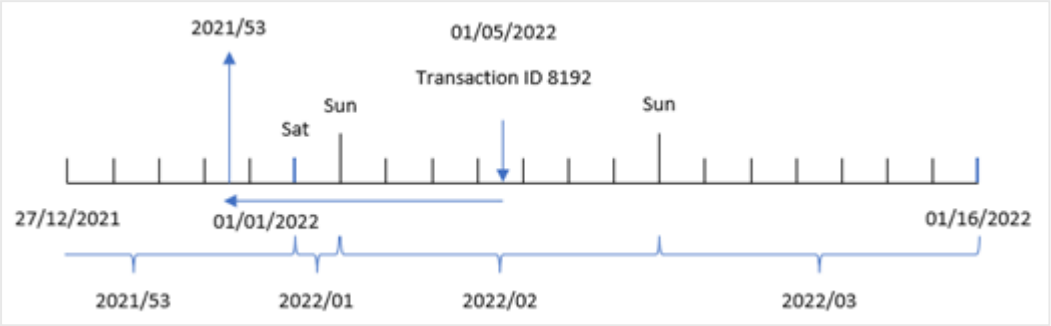
- id
- date
- week_day
- week_number

결과 테이블

id	date	week_day	week_number
8183	12/27/2021	Mon	2021/52
8184	12/28/2021	Tue	2021/52
8185	12/29/2021	Wed	2021/52
8186	12/30/2021	Thu	2021/52
8187	12/31/2021	Fri	2021/52
8188	01/01/2022	Sat	2021/52
8189	01/02/2022	Sun	2021/53
8190	01/03/2022	Mon	2021/53
8191	01/04/2022	Tue	2021/53
8192	01/05/2022	Wed	2021/53
8193	01/06/2022	Thu	2021/53
8194	01/07/2022	Fri	2021/53
8195	01/08/2022	Sat	2022/01
8196	01/09/2022	Sun	2022/02
8197	01/10/2022	Mon	2022/02
8198	01/11/2022	Tue	2022/02
8199	01/12/2022	Wed	2022/02
8200	01/13/2022	Thu	2022/02
8201	01/14/2022	Fri	2022/02

period_no -1은 weekname() 함수에서 오프셋 인수로 사용되므로 함수는 먼저 트랜잭션이 발생한 주를 식별합니다. 그런 다음 1주일 전을 찾아 해당 주의 첫 번째 밀리초를 식별합니다.

period_no 오프셋이 -1인 weekname() 함수 다이어그램.



트랜잭션 8192는 2022년 1월 5일에 발생했습니다. weekname() 함수는 1주일 전인 2021년 12월 30일을 찾아 해당 날짜의 주 수와 연도를 반환합니다(2021/53).

예 3 – first_week_day

로드 스크립트 및 결과

개요

첫 번째 예와 동일한 데이터 집합 및 시나리오가 사용됩니다.

그러나 이 예에서 회사 정책은 근무 주를 화요일에 시작하는 것입니다.

데이터 로드 편집기를 열고 다음의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

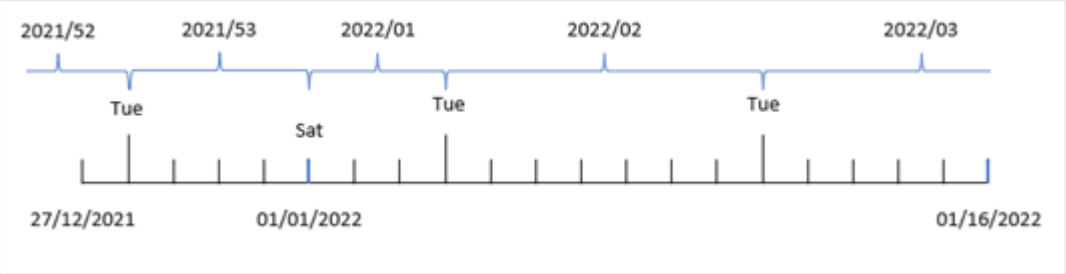
- id
- date
- week_day
- week_number

결과 테이블

id	date	week_day	week_number
8183	12/27/2021	Mon	2021/52
8184	12/28/2021	Tue	2021/53
8185	12/29/2021	Wed	2021/53
8186	12/30/2021	Thu	2021/53
8187	12/31/2021	Fri	2021/53
8188	01/01/2022	Sat	2022/01
8189	01/02/2022	Sun	2022/01
8190	01/03/2022	Mon	2022/01
8191	01/04/2022	Tue	2022/02
8192	01/05/2022	Wed	2022/02
8193	01/06/2022	Thu	2022/02
8194	01/07/2022	Fri	2022/02
8195	01/08/2022	Sat	2022/02
8196	01/09/2022	Sun	2022/02
8197	01/10/2022	Mon	2022/02

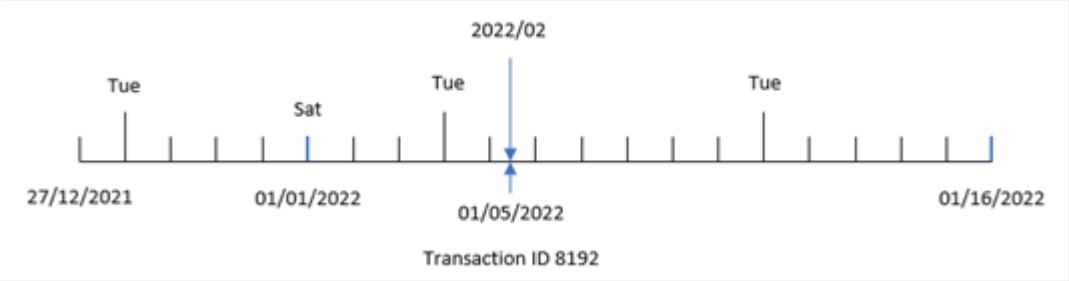
id	date	week_day	week_number
8198	01/11/2022	Tue	2022/03
8199	01/12/2022	Wed	2022/03
8200	01/13/2022	Thu	2022/03
8201	01/14/2022	Fri	2022/03

화요일을 주의 첫 번째 요일로 하는 weekname() 함수 다이어그램.



weekname() 함수에서 first_week_date 인수 1을 사용하므로 화요일을 주의 첫 번째 요일로 사용합니다. 따라서 이 함수는 2021년 53주가 12월 28일 화요일에 시작한다고 결정합니다. 그리고 분리된 주를 사용하는 응용 프로그램으로 인해 1주차는 2022년 1월 1일에 시작하여 2022년 1월 3일 월요일의 마지막 밀리초에 끝납니다.

화요일이 주의 첫 번째 요일인 트랜잭션 8192의 주 수를 보여 주는 다이어그램.



트랜잭션 8192는 2022년 1월 5일에 발생했습니다. 따라서 화요일의 first_week_day 매개 변수를 사용하여 weekname() 함수는 'week_number' 필드의 값 2022/02를 반환합니다.

예 4 – 차트 개체 예

로드 스크립트 및 차트 표현식

개요

첫 번째 예와 동일한 데이터 집합 및 시나리오가 사용됩니다.

그러나 이 예에서 데이터 집합은 변경되지 않고 응용 프로그램에 로드됩니다. 트랜잭션이 발생한 연도의 주 수를 반환하는 계산은 응용 프로그램의 차트 개체에서 측정값으로 만들어집니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- id
- date
- =week_day (date)

트랜잭션이 발생한 주의 시작을 계산하려면 다음 측정값을 만듭니다.

```
=weekname(date)
```

결과 테이블

id	date	=weekday(date)	=weekname(date)
8183	12/27/2021	Mon	2021/53
8184	12/28/2021	Tue	2021/53
8185	12/29/2021	Wed	2021/53
8186	12/30/2021	Thu	2021/53

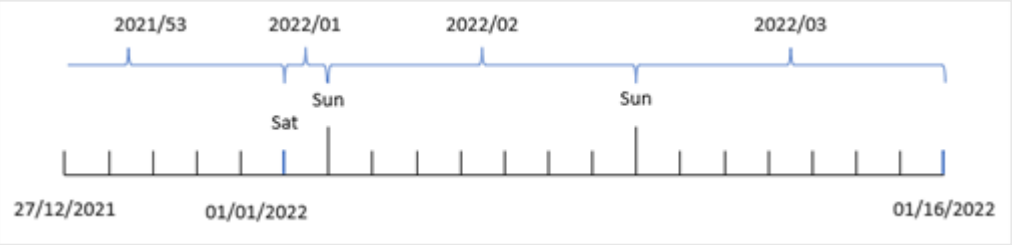
id	date	=weekday(date)	=weekname(date)
8187	12/31/2021	Fri	2021/53
8188	01/01/2022	Sat	2022/01
8189	01/02/2022	Sun	2022/02
8190	01/03/2022	Mon	2022/02
8191	01/04/2022	Tue	2022/02
8192	01/05/2022	Wed	2022/02
8193	01/06/2022	Thu	2022/02
8194	01/07/2022	Fri	2022/02
8195	01/08/2022	Sat	2022/02
8196	01/09/2022	Sun	2022/03
8197	01/10/2022	Mon	2022/03
8198	01/11/2022	Tue	2022/03
8199	01/12/2022	Wed	2022/03
8200	01/13/2022	Thu	2022/03
8201	01/14/2022	Fri	2022/03

'week_number' 필드는 weekname() 함수를 사용하고 함수의 인수로 날짜 필드를 전달하여 차트 개체에서 측정값으로 만들어집니다.

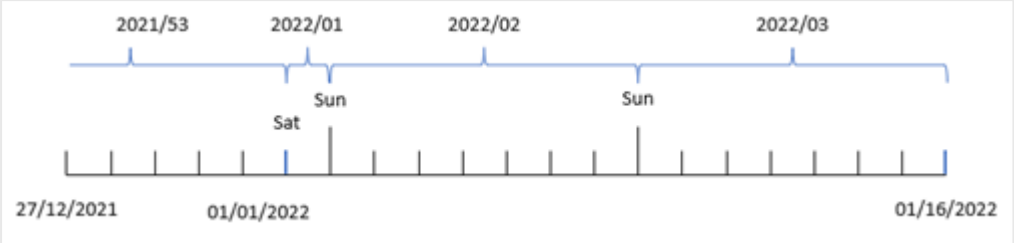
weekname() 함수는 처음에 날짜 값이 속하는 주를 식별하고 트랜잭션이 발생한 연도와 주 수 개수를 반환합니다.

FirstWeekDay 시스템 변수는 일요일을 주의 첫 번째 요일로 설정합니다. BrokenWeeks 시스템 변수는 분리된 주를 사용하도록 응용 프로그램을 설정합니다. 즉, 1주차는 1월 1일에 시작됩니다.

일요일이 주의 첫 번째 요일인 주 수를 보여 주는 다이어그램.



트랜잭션 8192가 두 번째 주에 발생했음을 보여 주는 다이어그램.



응용 프로그램이 분리된 주를 사용하고 첫 번째 요일이 일요일이기 때문에 1월 2일부터 1월 8일까지 발생한 트랜잭션은 값 2022/02(2022년의 두 번째 주)를 반환합니다. 트랜잭션 8192는 1월 5일에 발생했으며 'week_number' 필드에 대한 값 2022/02를 반환합니다.

예 5 – 시나리오

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 'Transactions'라는 테이블에 로드되는 2019년의 마지막 주와 2020년의 처음 2주 동안의 트랜잭션 집합을 포함하는 데이터 집합.
- 0으로 설정된 BrokenWeeks 시스템 변수.
- 2로 설정된 ReferenceDay 시스템 변수.
- MM/DD/YYYY 서식으로 설정된 DateFormat 시스템 변수.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만듭니다.

다음 표현식을 사용하여 계산된 차원을 만듭니다.

```
=weekname(date)
```

총 판매액을 계산하려면 다음 집계 측정값을 만듭니다.

```
=sum(amount)
```

측정값의 **숫자 형식**을 **화폐**로 설정합니다.

결과 테이블

weekname(date)	=sum(amount)
2019/52	$125.69
2020/01	$346.51
2020/02	$347.57
2020/03	$122.01

이 시나리오에서 weekname() 함수를 사용한 결과를 보여 주기 위해 다음 필드를 차원으로 추가합니다.

```
date
```

날짜 필드가 있는 결과 테이블

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
2020/01	01/02/2020	$17.17

weekname(date)	date	=sum(amount)
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

응용 프로그램은 분리되지 않은 주를 사용하고 ReferenceDay 시스템 변수로 인해 1주차에는 1월에 최소 2일이 필요하므로 2020년 1주차에는 2019년 12월 29일의 트랜잭션이 포함됩니다.

weekstart

이 함수는 **date**를 포함하는 캘린더 주의 첫 번째 날의 첫 번째 밀리초의 타임스탬프에 해당하는 값을 반환합니다. 기본 출력 형식은 스크립트에 설정된 **DateFormat**입니다.

구문:

```
WeekStart(timestamp [, period_no [, first_week_day ]])
```

반환 데이터 유형: dual

weekstart() 함수는 날짜가 속하는 주를 결정합니다. 그런 다음 해당 주의 첫 번째 밀리초에 대한 타임스탬프를 날짜 서식으로 반환합니다. 주의 첫 번째 요일은 FirstWeekDay 환경 변수에 의해 결정됩니다. 그러나 이는 weekstart() 함수의 first_week_day 인수로 대체될 수 있습니다.

인수

인수	설명
timestamp	평가할 날짜 또는 타임스탬프입니다.
period_no	**shift**는 정수이며, 값 0은 **date**를 포함하는 주를 나타냅니다. shift가 음수 값일 경우 이전 주, 양수 값일 경우 다음 주를 나타냅니다.

인수	설명
first_week_day	주의 시작 요일을 지정합니다. 생략하면 변수 **FirstWeekDay**의 값이 사용됩니다. **first_week_day**에 가능한 값은 월요일에 0, 화요일에 1, 수요일에 2, 목요일에 3, 금요일에 4, 토요일에 5, 일요일에 6입니다. 시스템 변수에 대한 자세한 내용은 *FirstWeekDay (page 219)*를 참조하십시오.

사용 시기

weekstart() 함수는 일반적으로 사용자가 지금까지 경과된 주의 부분을 사용하여 계산하려고 할 때 표현식의 일부로 사용됩니다. 예를 들어, 사용자가 지금까지 한 주 동안 직원이 받은 총 급여를 계산하려는 경우 사용할 수 있습니다.

다음 예에서는 다음을 가정합니다.

```
SET FirstWeekDay=0;
```

함수 예

예	결과
weekstart('01/12/2013')	01/07/2013를 반환합니다.
weekstart('01/12/2013', -1)	11/31/2012를 반환합니다.
weekstart('01/12/2013', 0, 1)	01/08/2013를 반환합니다.

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

주 및 주차에 대해 ISO 설정을 사용하려는 경우 스크립트에 다음이 포함되어 있는지 확인합니다.

```
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
```

미국 설정을 사용하려는 경우 스크립트에 다음이 포함되어 있는지 확인합니다.

```
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
```

위의 예는 weekstart() 함수에서 다음과 같은 결과를 가져옵니다.

Weekstart 함수의 예

Date	ISO 주 시작	US 주 시작
Sat 2020 Dec 26	2020-12-21	12/20/2020
Sun 2020 Dec 27	2020-12-21	12/27/2020
Mon 2020 Dec 28	2020-12-28	12/27/2020
Tue 2020 Dec 29	2020-12-28	12/27/2020
Wed 2020 Dec 30	2020-12-28	12/27/2020
Thu 2020 Dec 31	2020-12-28	12/27/2020
Fri 2021 Jan 1	2020-12-28	12/27/2020
Sat 2021 Jan 2	2020-12-28	12/27/2020
Sun 2021 Jan 3	2020-12-28	1/3/2021
Mon 2021 Jan 4	2021-01-04	1/3/2021
Tue 2021 Jan 5	2021-01-04	1/3/2021

주 시작은 ISO 열에서 월요일이고 미국 열에서 일요일입니다.

예 1 – 추가 인수 없음

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Transactions라는 테이블에 로드되는 2022년 트랜잭션 집합이 포함된 데이터 집합.
- DateFormat 시스템 변수(MM/DD/YYYY) 서식으로 제공된 날짜 필드.
- 트랜잭션이 발생한 주의 시작에 대한 타임스탬프를 반환하는 필드 start_of_week 만들기.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- start_of_week
- start_of_week_timestamp

결과 테이블

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

date	start_of_week	start_of_week_timestamp
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

start_of_week 필드는 weekstart() 함수를 사용하고 날짜 필드를 함수의 인수로 전달하여 선행 LOAD 문에서 만들어집니다.

weekstart() 함수는 날짜 값이 속하는 주를 초기에 식별하고 해당 주의 첫 번째 밀리초에 대한 타임스탬프를 반환합니다.

weekstart() 함수 다이어그램, 추가 인수가 없는 예

트랜잭션 8191은 2월 5일에 발생했습니다. FirstWeekDay 시스템 변수는 주의 첫 번째 요일을 일요일로 설정합니다. weekstart() 함수는 2월 5일 이전의 첫 번째 일요일(따라서 주의 시작)이 1월 30일임을 식별합니다. 따라서 해당 트랜잭션의 start_of_week 값은 해당 날짜의 첫 번째 밀리초, 즉 1월 30일 오전 12:00:00를 반환합니다.

예 2 – period_no

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 첫 번째 예와 동일한 데이터 집합 및 시나리오.
- 트랜잭션이 발생하기 전 분기 시작에 대한 타임스탬프를 반환하는 필드 previous_week_start 만들기.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

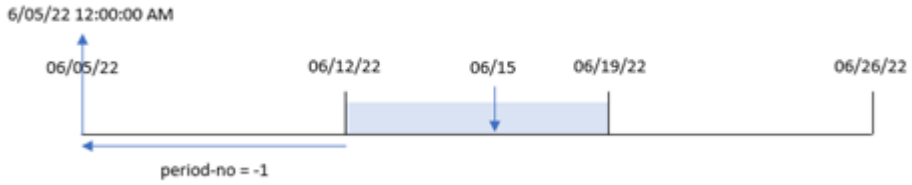
- date
- previous_week_start
- previous_week_start_timestamp

결과 테이블

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

이 경우 weekstart() 함수에서 period_no -1을 오프셋 인수로 사용했으므로 이 함수는 먼저 트랜잭션이 발생한 주를 식별합니다. 그런 다음 1주일 전을 찾아 해당 주의 첫 번째 밀리초를 식별합니다.

weekstart() 함수의 다이어그램, period_no 예



트랜잭션 8196은 6월 15일에 발생했습니다. weekstart() 함수는 주가 6월 12일에 시작함을 식별합니다. 따라서 이전 주는 6월 5일 오전 12:00:00에 시작되었습니다. previous_week_start 필드에 대해 반환되는 값입니다.

예 3 – first_week_day

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

이 로드 스크립트에는 첫 번째 예와 동일한 데이터 집합 및 시나리오가 포함되어 있습니다. 그러나 이 예에서는 화요일을 근무 주의 첫 번째 요일로 설정해야 합니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- start_of_week
- start_of_week_timestamp

결과 테이블

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
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

이 경우 weekstart() 함수에서 first_week_date 인수 1을 사용하므로 주의 첫 번째 날을 화요일로 설정합니다.

weekstart() 함수 다이어그램, first_week_day 예

트랜잭션 8191은 2월 5일에 발생했습니다. weekstart() 함수는 이 날짜 이전의 첫 번째 화요일(따라서 주의 시작 및 반환된 값)이 2월 1일 오전 12:00:00임을 식별합니다.

예 4 – 차트 개체 예

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

이 로드 스크립트에는 첫 번째 예와 동일한 데이터 집합 및 시나리오가 포함되어 있습니다.

그러나 이 예에서는 변경되지 않은 데이터 집합이 응용 프로그램에 로드됩니다. 트랜잭션이 발생한 주의 시작에 대한 타임스탬프를 반환하는 계산은 응용 프로그램의 차트 개체에서 측정값으로 만들어집니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다. date.

트랜잭션이 발생한 주의 시작을 계산하려면 다음 측정값을 추가합니다.

- =weekstart(date)
- =timestamp(weekstart(date))

결과 테이블

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

`start_of_week` 측정값은 `weekstart()` 함수를 사용하고 `date` 필드를 함수의 인수로 전달하여 차트 개체에서 만들어집니다.

`weekstart()` 함수는 날짜 값이 속하는 주를 초기에 식별하고 해당 주의 첫 번째 밀리초에 대한 타임스탬프를 반환합니다.

weekstart() 함수의 다이어그램, 차트 개체 예

트랜잭션 8191은 2월 5일에 발생했습니다. FirstWeekDay 시스템 변수는 주의 첫 번째 요일을 일요일로 설정합니다. weekstart() 함수는 2월 5일 이전의 첫 번째 일요일(따라서 주의 시작)이 1월 30일임을 식별합니다. 따라서 해당 트랜잭션의 start_of_week 값은 해당 날짜의 첫 번째 밀리초, 즉 1월 30일 오전 12:00:00를 반환합니다.

예 5 – 시나리오

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Payroll라는 테이블에 로드되는 데이터 집합.
- 직원 ID, 직원 이름, 직원별 일일 급여로 구성된 데이터.

직원들은 월요일에 근무를 시작하여 주 6일 근무합니다. FirstWeekDay 시스템 변수는 수정하면 안 됩니다.

최종 사용자는 직원 ID 및 직원 이름별로 해당 주부터 현재까지 번 임금을 표시하는 차트 개체를 원합니다.

로드 스크립트

```
Payroll:
Load
*
Inline
[
employee_id,employee_name,day_rate
182,Mark, $150
183,Deryck, $125
184,Dexter, $125
185,Sydney,$270
186,Agatha,$128
];
```

결과

다음과 같이 하십시오.

1. 데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.
   - employee_id
   - employee_name
2. 다음으로 해당 주부터 현재까지 번 임금을 계산하는 측정값을 만듭니다.
   `=if(today(1)-weekstart(today(1),0,0)<7,(today(1)-weekstart(today(1),0,0))*day_rate,day_rate*6)`
3. 측정값의 **숫자 형식**을 **화폐**로 설정합니다.

결과 테이블

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

weekstart() 함수는 오늘 날짜를 첫 번째 인수로 사용하고 0을 세 번째 인수로 사용하여 월요일을 주의 첫 번째 요일로 설정하고 현재 주의 시작 날짜를 반환합니다. 현재 날짜에서 해당 결과를 빼면 이 표현식은 이번 주에 지금까지 경과한 일 수를 반환합니다.

그런 다음 이 조건은 이번 주에 6일보다 많이 있었는지 여부를 평가합니다. 그렇다면 직원의 day_rate에 6일을 곱합니다. 그렇지 않으면 day_rate에 이번 주에 지금까지 발생한 일 수를 곱합니다.

weekyear

이 함수는 환경 변수에 따라 해당 주차가 속한 연도를 반환합니다. 주차 범위는 1과 약 52 사이입니다.

구문:

```
weekyear(timestamp [, first_week_day [, broken_weeks [, reference_day]]])
```

반환 데이터 유형: 정수

인수

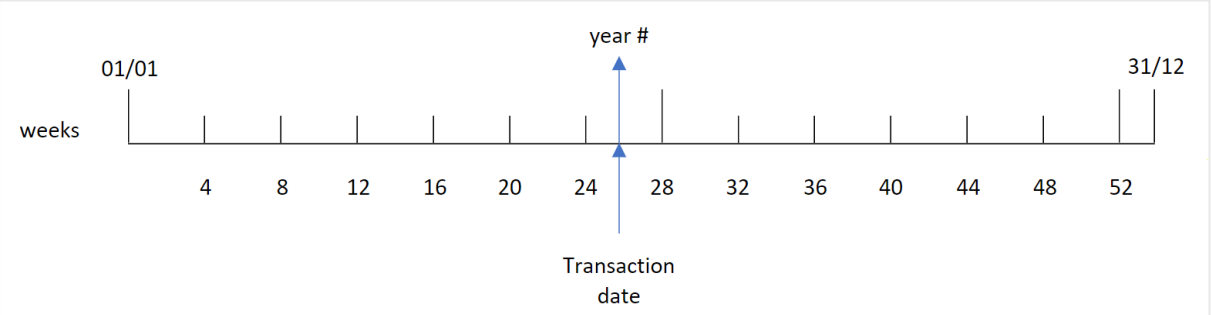
인수	설명
timestamp	평가할 날짜 또는 타임스탬프입니다.

인수	설명
first_week_day	주의 시작 요일을 지정합니다. 생략하면 변수 **FirstWeekDay**의 값이 사용됩니다. **first_week_day**에 가능한 값은 월요일에 0, 화요일에 1, 수요일에 2, 목요일에 3, 금요일에 4, 토요일에 5, 일요일에 6입니다. 시스템 변수에 대한 자세한 내용은 *FirstWeekDay (page 219)*를 참조하십시오.
broken_weeks	**broken_weeks**를 지정하지 않으면 **BrokenWeeks** 변수의 값이 주를 분리할지 여부를 정의하는 데 사용됩니다.
reference_day	**reference_day**를 지정하지 않으면 **ReferenceDay** 변수의 값이 1주차를 정의하기 위한 기준일로 설정할 1월 날짜를 정의하는 데 사용됩니다. 기본적으로 Qlik Sense 함수는 4를 기준일로 사용합니다. 이는 1주차에 1월 4일이 포함되어야 한다는 의미입니다. 혹은 달리 말하면 1주차에 항상 1월이 4일 이상이 있어야 합니다.

weekyear() 함수는 날짜가 속하는 연도의 주를 결정합니다. 그런 다음 해당 주차에 해당하는 연도를 반환합니다.

BrokenWeeks가 0(false)으로 설정되면 weekyear()는 year()와 동일하게 반환됩니다.

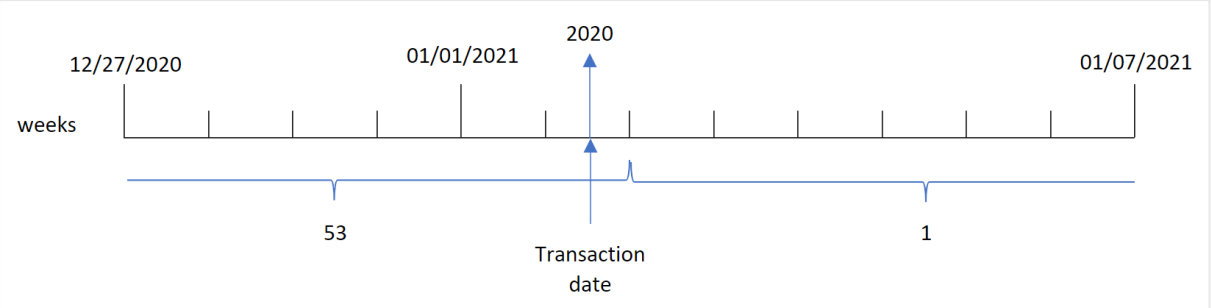
weekyear() 함수의 범위 다이어그램



그러나 BrokenWeeks 시스템 변수가 분리되지 않은 주를 사용하도록 설정된 경우 1주차는 ReferenceDay 시스템 변수에 지정된 값을 기준으로 1월의 특정 일 수만 포함해야 합니다.

예를 들어 ReferenceDay 값 4가 사용되는 경우 1주차에는 1월의 4일이 포함되어야 합니다. 1주차에는 전년도 12월의 날짜가 포함되거나 연도의 마지막 주차에 다음 해 1월의 날짜가 포함될 수 있습니다. 이와 같은 상황에서 weekyear() 함수는 year() 함수에 다른 값을 반환합니다.

분리되지 않은 주를 사용할 때의 weekyear() 함수 범위의 다이어그램



사용 시기

weekyear() 함수는 연도별로 집계를 비교하려는 경우에 유용합니다. 예를 들어 연도별 제품의 총 판매액을 보려는 경우에 사용할 수 있습니다. 사용자가 앱의 BrokenWeeks 시스템 변수와 일관성을 유지하기를 원할 때 year()보다는 weekyear() 함수가 선택됩니다.

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

함수 예

예	결과
weekyear ('12/30/1996',0,0,4)	1997년의 1주차가 1996년 12월 30일에 시작되므로 1997년을 반환합니다.
weekyear ('01/02/1997',0,0,4)	1997을 반환합니다.
weekyear ('12/28/1997',0,0,4)	1997을 반환합니다.
weekyear ('12/30/1997',0,0,4)	1998년의 1주차가 1997년 12월 29일에 시작되므로 1998년을 반환합니다
weekyear ('01/02/1999',0,0,4)	1998년 53주차가 1999년 1월 3일에 끝나므로 1998년을 반환합니다.

관련 항목

항목	상호 작용
week (page 1015)	ISO 8601에 따라 주차를 나타내는 정수를 반환합니다.
year (page 1087)	이 표현식이 표준 숫자 해석에 따라 날짜로 해석될 때 연도를 나타내는 정수를 반환합니다.

예 1 - 분리된 주

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 'Transactions'라는 테이블에 로드되는 2020년 마지막 주와 2021년 첫 주에 대한 트랜잭션 집합을 포함하는 데이터 집합.
- 1로 설정된 BrokenWeeks 변수.
- 다음을 포함하는 선행 LOAD:
    - weekyear() 함수, 트랜잭션이 발생한 연도를 반환하는 필드 'week_year'로 설정됩니다.
    - week() 함수, 각 트랜잭션 날짜의 주차를 표시하는 필드 'week'로 설정됩니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- id
- date
- week
- week_year

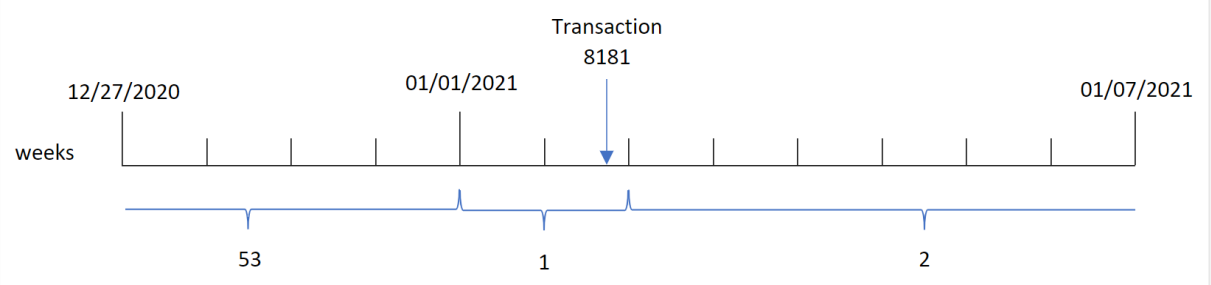
결과 테이블

id	date	주	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

'week_year' 필드는 weekyear() 함수를 사용하고 날짜 필드를 함수의 인수로 전달하여 선행 LOAD 문에서 만들어집니다.

BrokenWeeks 시스템 변수는 앱이 분리된 주를 사용함을 의미하는 1로 설정됩니다. 1주차가 1월 1일에 시작됩니다.

분리된 주를 사용하는 weekyear() 함수 범위의 다이어그램



트랜잭션 8181은 1주차에 속하는 1월 2일에 발생합니다. 따라서 'week_year' 필드에 대해 값 2021을 반환합니다.

예 2 - 분리되지 않은 주

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 'Transactions'라는 테이블에 로드되는 2020년 마지막 주와 2021년 첫 주에 대한 트랜잭션 집합을 포함하는 데이터 집합.
- 0으로 설정된 BrokenWeeks 변수.
- 다음을 포함하는 선행 LOAD:
    - weekyear() 함수, 트랜잭션이 발생한 연도를 반환하는 필드 'week_year'로 설정됩니다.
    - week() 함수, 각 트랜잭션 날짜의 주차를 표시하는 필드 'week'로 설정됩니다.

그러나 이 예에서 회사 정책은 분리되지 않은 주를 사용하는 것입니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- id
- date
- week
- week_year

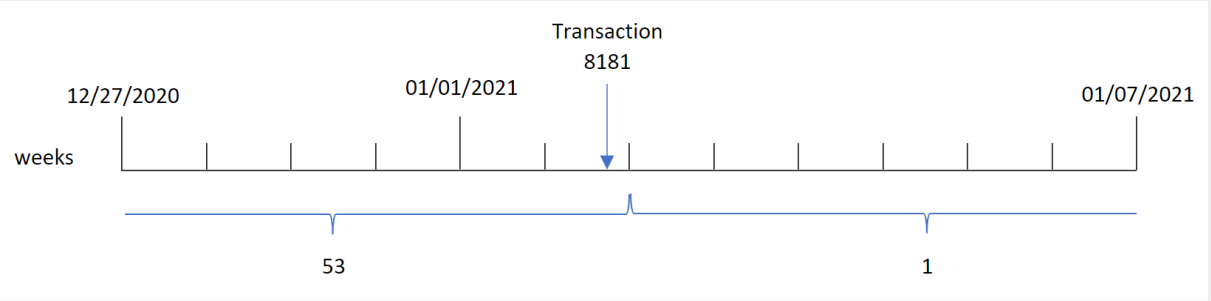
결과 테이블

id	date	주	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

BrokenWeeks 시스템 변수는 응용 프로그램이 분리된 주를 사용함을 의미하는 0으로 설정됩니다. 따라서 1주차는 1월 1일에 시작할 필요가 없습니다.

2020년의 53주차는 2021년 1월 2일까지 계속되고 2021년의 1주차는 2021년 1월 3일 일요일에 시작됩니다.

분리되지 않은 주를 사용하는 weekyear() 함수 범위의 다이어그램



트랜잭션 8181은 1주차에 속하는 1월 2일에 발생합니다. 따라서 'week_year' 필드에 대해 값 2021을 반환합니다.

예 3 - 차트 개체 예

로드 스크립트 및 차트 표현식

개요

첫 번째 예와 동일한 데이터 집합 및 시나리오가 사용됩니다.

그러나 이 예에서 데이터 집합은 변경되지 않고 응용 프로그램에 로드됩니다. 트랜잭션이 발생한 연도의 주차를 반환하는 계산은 앱의 차트에서 측정값으로 만들어집니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- id
- date

트랜잭션이 발생한 주를 계산하려면 다음 측정값을 만듭니다.

- =week(date)

주차를 기준으로 트랜잭션이 발생한 연도를 계산하려면 다음 측정값을 만듭니다.

- =weekyear(date)

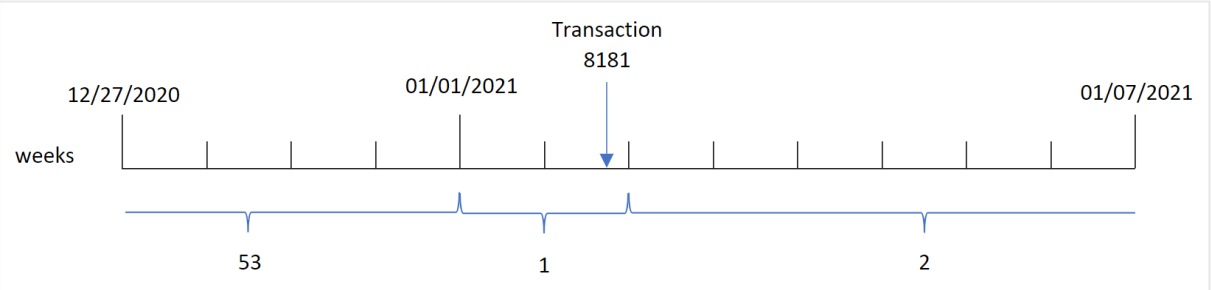
결과 테이블

id	date	주	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

'week_year' 필드는 weekyear() 함수를 사용하고 날짜 필드를 함수의 인수로 전달하여 선행 LOAD 문에서 만들어집니다.

BrokenWeeks 시스템 변수는 앱이 분리된 주를 사용함을 의미하는 1로 설정됩니다. 1주차가 1월 1일에 시작됩니다.

분리된 주를 사용하는 weekyear() 함수 범위의 다이어그램



트랜잭션 8181은 1주차에 속하는 1월 2일에 발생합니다. 따라서 'week_year' 필드에 대해 값 2021을 반환합니다.

예 4 - 시나리오

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 'Transactions'라는 테이블에 로드되는 2020년 마지막 주와 2021년 첫 주에 대한 트랜잭션 집합을 포함하는 데이터 집합.
- 0으로 설정된 BrokenWeeks 변수. 이는 앱이 분리되지 않은 주를 사용한다는 것을 의미합니다.
- 2으로 설정된 ReferenceDay 변수. 이는 연도가 1월 2일에 시작되고 1월에 최소 2일이 포함됨을 의미합니다.
- 1으로 설정된 FirstWeekDay 변수. 이는 주의 첫 번째 날이 화요일임을 의미합니다.

회사 정책은 분리된 주를 사용하는 것입니다. 최종 사용자는 연도별 총 판매를 나타내는 차트를 원합니다. 이 앱은 1월에 최소 2일이 포함된 1주차와 함께 분리되지 않은 주를 사용합니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만듭니다.

주차를 기준으로 트랜잭션이 발생한 연도를 계산하려면 다음 측정값을 만듭니다.

- =weekyear(date)

총 판매액을 계산하려면 다음 측정값을 만듭니다.

- sum(amount)

측정값의 **숫자 형식**을 **화폐**로 설정합니다.

결과 테이블

weekyear(date)	=sum(amount)
2020	19.42
2021	373.37

year

이 함수는 **expression**이 표준 숫자 해석에 따라 날짜로 해석될 경우 연도를 나타내는 정수를 반환합니다.

구문:

```
year(expression)
```

반환 데이터 유형: 정수

year() 함수는 스크립트 및 차트 함수로 사용할 수 있습니다. 이 함수는 특정 날짜의 연도를 반환합니다. 일반적으로 마스터 캘린더의 차원으로 연도 필드를 만드는 데 사용됩니다.

사용 시기

year() 함수는 연도별로 집계를 비교하려는 경우에 유용합니다. 예를 들어 연도별 제품의 총 판매액을 보고 싶을 때 이 함수를 사용할 수 있습니다.

이러한 차원은 마스터 캘린더 테이블에 필드를 만드는 함수를 사용하여 로드 스크립트에서 만들 수 있습니다. 또는 차트에서 계산된 차원으로 직접 사용할 수 있습니다.

함수 예

예	결과
year('2012-10-12')	2012를 반환합니다.
year('35648')	35648 = 1997-08-06이므로, 1997을 반환합니다.

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

예 1 – DateFormat 데이터 집합(스크립트)

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Master Calendar라는 테이블에 로드되는 날짜 데이터 집합.
- 기본 DateFormat 시스템 변수 MM/DD/YYYY가 사용됩니다.
- year() 함수를 사용하여 추가 필드 year를 만드는 데 사용되는 선행 LOAD.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- year

결과 테이블

date	년
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

예 2 - ANSI 날짜

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- `Master Calendar`라는 테이블에 로드되는 날짜 데이터 집합.
- 기본 `DateFormat` 시스템 변수(MM/DD/YYYY)가 사용됩니다. 그러나 데이터 집합에 포함된 날짜는 ANSI 표준 날짜 서식입니다.
- `year()` 함수를 사용하여 'year'라는 추가 필드를 만드는 데 사용되는 선행 LOAD.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- year

결과 테이블

date	년
2020-12-28	2020
2020-12-29	2020
2020-12-30	2020
2020-12-31	2020
2021-01-01	2021
2021-01-02	2021
2021-01-03	2021
2021-01-04	2021
2021-01-05	2021
2021-01-06	2021
2021-01-07	2021

예 3 – 서식이 지정되지 않은 날짜

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Master Calendar라는 테이블에 로드되는 숫자 서식의 날짜 데이터 집합.
- 기본 DateFormat 시스템 변수(MM/DD/YYYY)가 사용됩니다.
- year() 함수를 사용하여 추가 필드 'year'를 만드는 데 사용되는 선행 LOAD.

서식이 지정되지 않은 원본 날짜(unformatted_date)가 로드되고 명확성을 제공하기 위해 date() 함수를 사용하여 숫자 날짜를 서식이 지정된 날짜 필드로 변환하는 데 long_date라는 추가 필드가 사용됩니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- unformatted_date
- long_date
- year

결과 테이블

unformatted_date	long_date	년
44868	11/03/2022	2022
44898	12/03/2022	2022
44928	01/02/2023	2023

unformatted_date	long_date	년
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

예 4 – 차트 개체 예

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

이 예에서는 주문 데이터 집합이 Sales라는 테이블에 로드됩니다. 테이블에는 세 개의 필드가 있습니다.

- id
- sales_date
- amount

제품 판매에 대한 보증은 판매일로부터 2년 동안 지속됩니다. 차트에 측정값을 만들어 각 보증이 만료되는 연도를 확인하는 작업입니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다. sales_date.

다음 측정값을 만듭니다.

```
=year(sales_date+365*2)
```

결과 테이블

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

이 측정값의 결과는 위의 테이블에서 볼 수 있습니다. 날짜에 2년을 더하려면 365에 2를 곱하고 그 결과를 판매 날짜에 더합니다. 따라서 2020년에 발생한 판매의 만료 연도는 2022년입니다.

yearend

이 함수는 **date**를 포함하는 연도의 마지막 날의 마지막 밀리초의 타임스탬프에 해당하는 값을 반환합니다. 기본 출력 형식은 스크립트에 설정된 **DateFormat**입니다.

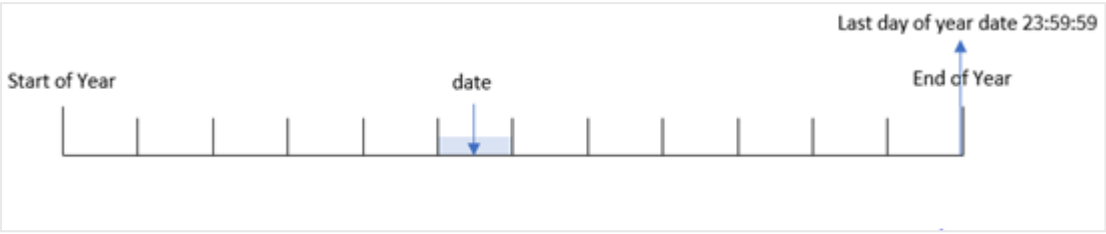
구문:

```
YearEnd( date[, period_no[, first_month_of_year = 1]])
```

즉, yearend() 함수는 이 날짜가 속하는 연도를 확인합니다. 그런 다음 해당 연도의 마지막 밀리초에 대한 타임스탬프를 날짜 서식으로 반환합니다. 연도의 첫 번째 달은 기본적으로 1월입니다. 그러나 yearend() 함수의 first_month_of_year 인수를 사용하여 첫 번째로 설정되는 월을 변경할 수 있습니다.

yearend() 함수는 FirstMonthOfYear 시스템 변수를 고려하지 않습니다. first_month_of_year 인수를 사용하여 변경하지 않는 한 연도는 1월 1일에 시작됩니다.

yearend() 함수 다이어그램.



사용 시기

yearend() 함수는 아직 발생하지 않은 연도의 분수를 사용하도록 계산하려는 경우 표현식의 일부로 사용됩니다. 예를 들어 해당 연도에 아직 발생하지 않은 총 이자를 계산하려는 경우가 있습니다.

반환 데이터 유형: dual

인수

인수	설명
date	평가할 날짜 또는 타임스탬프입니다.
period_no	**period_no**는 정수이며, 값 0은 **date**를 포함하는 연도를 나타냅니다. **period_no**가 음수 값일 경우 이전 연도, 양수 값일 경우 다음 연도를 나타냅니다.
first_month_of_year	1월에 시작되지 않는 (회계)연도를 사용하려는 경우 **first_month_of_year**에 2와 12 사이의 값을 지정하십시오.

다음 값을 사용하여 first_month_of_year 인수에서 연도의 첫 번째 달을 설정할 수 있습니다.

first_month_of_year 값

Month	Value
2월	2
3월	3
4월	4
5월	5
6월	6
7월	7
8월	8
9월	9
10월	10
11월	11
12월	12

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

함수 예

예	결과
yearend('10/19/2001')	12/31/2001 23:59:59를 반환합니다.
yearend('10/19/2001', -1)	12/31/2000 23:59:59를 반환합니다.
yearend('10/19/2001', 0, 4)	03/31/2002 23:59:59를 반환합니다.

예 1 – 추가 인수 없음

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 'Transactions'라는 테이블에 로드되는 2020년과 2022년 사이의 트랜잭션 집합이 포함된 데이터 집합.
- DateFormat 시스템 변수 서식((MM/DD/YYYY))으로 제공된 날짜 필드.
- 다음을 포함하는 선행 LOAD 문:
    - year_end 필드로 설정되는 yearend() 함수.
    - year_end_timestamp 필드로 설정되는 Timestamp() 함수.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- id
- date
- year_end
- year_end_timestamp

결과 테이블

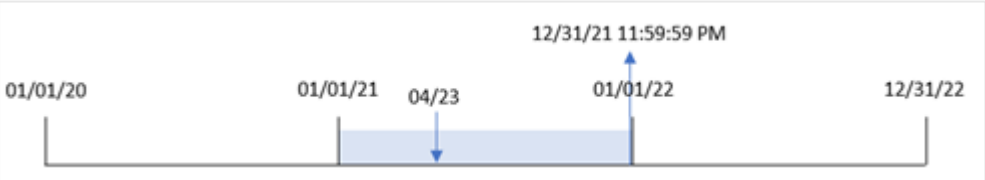
id	date	year_end	year_end_timestamp
8188	01/13/2020	12/31/2020	12/31/2020 11:59:59 PM
8189	02/26/2020	12/31/2020	12/31/2020 11:59:59 PM
8190	03/27/2020	12/31/2020	12/31/2020 11:59:59 PM
8191	04/16/2020	12/31/2020	12/31/2020 11:59:59 PM
8192	05/21/2020	12/31/2020	12/31/2020 11:59:59 PM
8193	08/14/2020	12/31/2020	12/31/2020 11:59:59 PM
8194	10/07/2020	12/31/2020	12/31/2020 11:59:59 PM

id	date	year_end	year_end_timestamp
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

'year_end' 필드는 yearend() 함수를 사용하고 날짜 필드를 함수의 인수로 전달하여 선행 LOAD 문에서 만들어집니다.

yearend() 함수는 날짜 값이 속하는 연도를 처음에 식별하고 해당 연도의 마지막 밀리초에 대한 타임스탬프를 반환합니다.

트랜잭션 8199가 선택된 yearend() 함수 다이어그램.



트랜잭션 8199는 2021년 4월 23일에 발생했습니다. yearend() 함수는 해당 연도의 마지막 밀리초, 즉 12월 31일 오후 11시 59분 59초를 반환합니다.

예 2 – period_no

로드 스크립트 및 결과

개요

첫 번째 예와 동일한 데이터 집합 및 시나리오가 사용됩니다.

그러나 이 예에서 작업은 트랜잭션이 발생한 연도 이전 연도의 종료 날짜 타임스탬프를 반환하는 'previous_year_end' 필드를 만드는 것입니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- id
- date
- previous_year_end
- previous_ year_end_timestamp

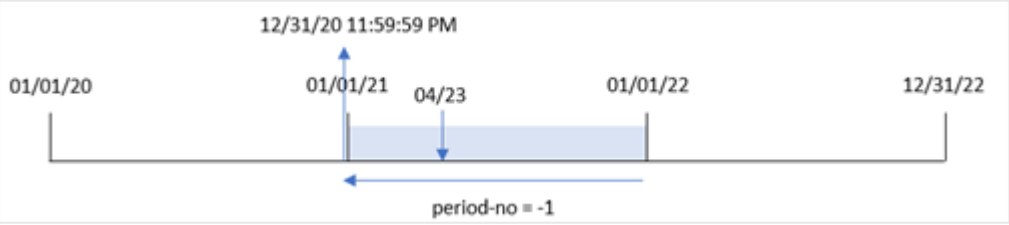
결과 테이블

id	date	previous_year_end	previous_year_end_timestamp
8188	01/13/2020	12/31/2019	12/31/2019 11:59:59 PM

id	date	previous_year_end	previous_year_end_timestamp
8189	02/26/2020	12/31/2019	12/31/2019 11:59:59 PM
8190	03/27/2020	12/31/2019	12/31/2019 11:59:59 PM
8191	04/16/2020	12/31/2019	12/31/2019 11:59:59 PM
8192	05/21/2020	12/31/2019	12/31/2019 11:59:59 PM
8193	08/14/2020	12/31/2019	12/31/2019 11:59:59 PM
8194	10/07/2020	12/31/2019	12/31/2019 11:59:59 PM
8195	12/05/2020	12/31/2019	12/31/2019 11:59:59 PM
8196	01/22/2021	12/31/2020	12/31/2020 11:59:59 PM
8197	02/03/2021	12/31/2020	12/31/2020 11:59:59 PM
8198	03/17/2021	12/31/2020	12/31/2020 11:59:59 PM
8199	04/23/2021	12/31/2020	12/31/2020 11:59:59 PM
8200	05/04/2021	12/31/2020	12/31/2020 11:59:59 PM
8201	06/30/2021	12/31/2020	12/31/2020 11:59:59 PM
8202	07/26/2021	12/31/2020	12/31/2020 11:59:59 PM
8203	12/27/2021	12/31/2020	12/31/2020 11:59:59 PM
8204	06/06/2022	12/31/2021	12/31/2021 11:59:59 PM
8205	07/18/2022	12/31/2021	12/31/2021 11:59:59 PM
8206	11/14/2022	12/31/2021	12/31/2021 11:59:59 PM
8207	12/12/2022	12/31/2021	12/31/2021 11:59:59 PM

period_no -1은 yearend() 함수에서 오프셋 인수로 사용되었으므로 함수는 먼저 트랜잭션이 발생한 연도를 식별합니다. 그런 다음 1년 전을 찾아 해당 연도의 마지막 밀리초를 식별합니다.

period_no가 -1인 yearend() 함수 다이어그램.



트랜잭션 8199는 2021년 4월 23일에 발생합니다. yearend() 함수는 'previous_year_end' 필드에 대해 전년도의 마지막 밀리초인 2020년 12월 31일 오후 11:59:59를 반환합니다.

예 3 – first_month_of_year

로드 스크립트 및 결과

개요

첫 번째 예와 동일한 데이터 집합 및 시나리오가 사용됩니다.

그러나 이 예에서는 회사 정책이 4월 1일부터 시작되는 연도를 기준으로 합니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

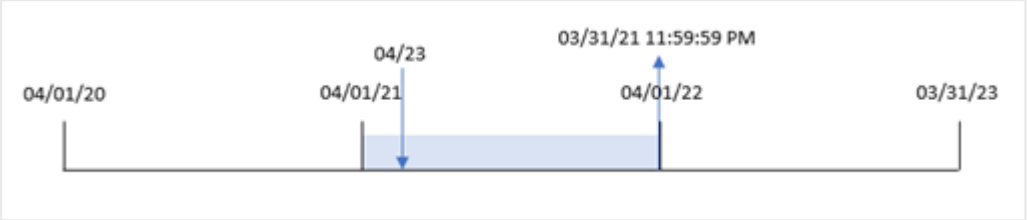
- id
- date
- year_end
- year_end_timestamp

결과 테이블

id	date	year_end	year_end_timestamp
8188	01/13/2020	03/31/2020	3/31/2020 11:59:59 PM
8189	02/26/2020	03/31/2020	3/31/2020 11:59:59 PM
8190	03/27/2020	03/31/2020	3/31/2020 11:59:59 PM
8191	04/16/2020	03/31/2021	3/31/2021 11:59:59 PM
8192	05/21/2020	03/31/2021	3/31/2021 11:59:59 PM
8193	08/14/2020	03/31/2021	3/31/2021 11:59:59 PM
8194	10/07/2020	03/31/2021	3/31/2021 11:59:59 PM
8195	12/05/2020	03/31/2021	3/31/2021 11:59:59 PM
8196	01/22/2021	03/31/2021	3/31/2021 11:59:59 PM
8197	02/03/2021	03/31/2021	3/31/2021 11:59:59 PM
8198	03/17/2021	03/31/2021	3/31/2021 11:59:59 PM
8199	04/23/2021	03/31/2022	3/31/2022 11:59:59 PM
8200	05/04/2021	03/31/2022	3/31/2022 11:59:59 PM
8201	06/30/2021	03/31/2022	3/31/2022 11:59:59 PM
8202	07/26/2021	03/31/2022	3/31/2022 11:59:59 PM
8203	12/27/2021	03/31/2022	3/31/2022 11:59:59 PM
8204	06/06/2022	03/31/2023	3/31/2023 11:59:59 PM
8205	07/18/2022	03/31/2023	3/31/2023 11:59:59 PM
8206	11/14/2022	03/31/2023	3/31/2023 11:59:59 PM
8207	12/12/2022	03/31/2023	3/31/2023 11:59:59 PM

yearend() 함수에서 first_month_of_year 인수 4을 사용하므로 해당 연도의 첫 번째 날이 4월 1일로 설정되고 해당 연도의 마지막 날이 3월 31일로 설정됩니다.

4월을 연도의 첫 번째 달로 사용하는 yearend() 함수 다이어그램.



트랜잭션 8199는 2021년 4월 23일에 발생합니다. yearend() 함수가 연도의 시작을 4월 1일로 설정하므로 2022년 3월 31일이 트랜잭션의 'year_end' 값으로 반환됩니다.

예 4 – 차트 개체 예

로드 스크립트 및 차트 표현식

개요

첫 번째 예와 동일한 데이터 집합 및 시나리오가 사용됩니다.

그러나 이 예에서 데이터 집합은 변경되지 않고 응용 프로그램에 로드됩니다. 트랜잭션이 발생한 연도의 끝 날짜 타임스탬프를 반환하는 계산은 응용 프로그램의 차트 개체에서 측정값으로 만들어집니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- id
- date

트랜잭션이 발생한 연도를 계산하려면 다음 측정값을 만듭니다.

- =yearend(date)
- =timestamp(yearend(date))

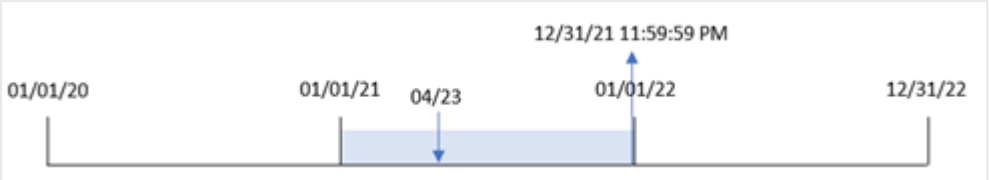
결과 테이블

id	date	=yearend(date)	=timestamp(yearend(date))
8188	01/13/2020	12/31/2020	12/31/2020 11:59:59 PM
8189	02/26/2020	12/31/2020	12/31/2020 11:59:59 PM
8190	03/27/2020	12/31/2020	12/31/2020 11:59:59 PM
8191	04/16/2020	12/31/2020	12/31/2020 11:59:59 PM
8192	05/21/2020	12/31/2020	12/31/2020 11:59:59 PM
8193	08/14/2020	12/31/2020	12/31/2020 11:59:59 PM
8194	10/07/2020	12/31/2020	12/31/2020 11:59:59 PM
8195	12/05/2020	12/31/2020	12/31/2020 11:59:59 PM
8196	01/22/2021	12/31/2021	12/31/2021 11:59:59 PM
8197	02/03/2021	12/31/2021	12/31/2021 11:59:59 PM
8198	03/17/2021	12/31/2021	12/31/2021 11:59:59 PM
8199	04/23/2021	12/31/2021	12/31/2021 11:59:59 PM
8200	05/04/2021	12/31/2021	12/31/2021 11:59:59 PM
8201	06/30/2021	12/31/2021	12/31/2021 11:59:59 PM
8202	07/26/2021	12/31/2021	12/31/2021 11:59:59 PM
8203	12/27/2021	12/31/2021	12/31/2021 11:59:59 PM
8204	06/06/2022	12/31/2022	12/31/2022 11:59:59 PM
8205	07/18/2022	12/31/2022	12/31/2022 11:59:59 PM
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

'end_of_year' 측정값은 yearend() 함수를 사용하고 날짜 필드를 함수의 인수로 전달하여 차트 개체에서 만들어집니다.

yearend() 함수는 날짜 값이 속하는 연도를 처음에 식별하고 해당 연도의 마지막 밀리초에 대한 타임스탬프를 반환합니다.

트랜잭션 8199가 4월에 발생했음을 보여 주는 yearend() 함수 다이어그램.



트랜잭션 8199는 2021년 4월 23일에 발생합니다. yearend() 함수는 해당 연도의 마지막 밀리초, 즉 12월 31일 오후 11시 59분 59초를 반환합니다.

예 5 – 시나리오

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 'Employee_Expenses'라는 테이블에 로드되는 데이터 집합. 테이블에는 다음 필드가 포함됩니다.
  - 직원 ID
  - 직원 이름
  - 각 직원의 평균 일일 비용 청구

최종 사용자는 직원 ID와 직원 이름별로 해당 연도에 남은 기간 동안 발생할 것으로 예상되는 비용 청구를 표시하는 차트 개체를 원합니다. 회계 연도는 1월에 시작됩니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- employee_id
- employee_name

예상 비용 청구를 계산하려면 다음 측정값을 만듭니다.

```
=(yearend(today(1))-today(1))*avg_daily_claim
```

측정값의 **숫자 형식**을 **화폐**로 설정합니다.

결과 테이블

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

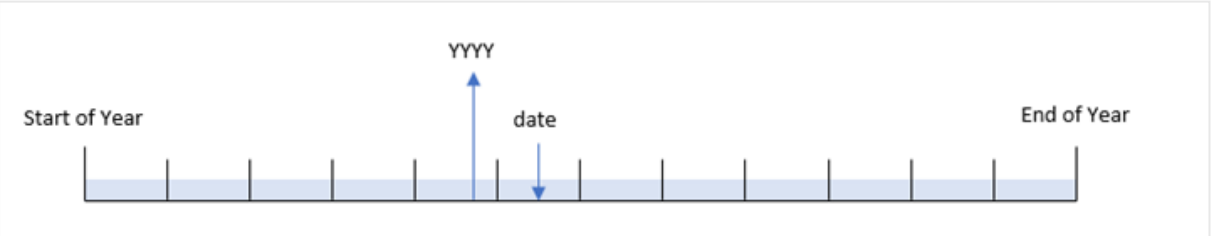
오늘 날짜를 유일한 인수로 사용하여 yearend() 함수는 현재 연도의 종료 날짜를 반환합니다. 그런 다음 연도 종료 날짜에서 오늘 날짜를 빼서 이 표현식은 올해의 남은 일수를 반환합니다.

그런 다음 이 값에 각 직원의 평균 일일 비용 청구를 곱하여 각 직원이 남은 연도에 예상되는 청구 금액을 계산합니다.

yearname

이 함수는 **date**를 포함한 연도의 첫 번째 날의 첫 번째 밀리초의 타임스탬프에 해당하는 기본 숫자 값을 사용하여 네 자리 연도를 표시 값으로 반환합니다.

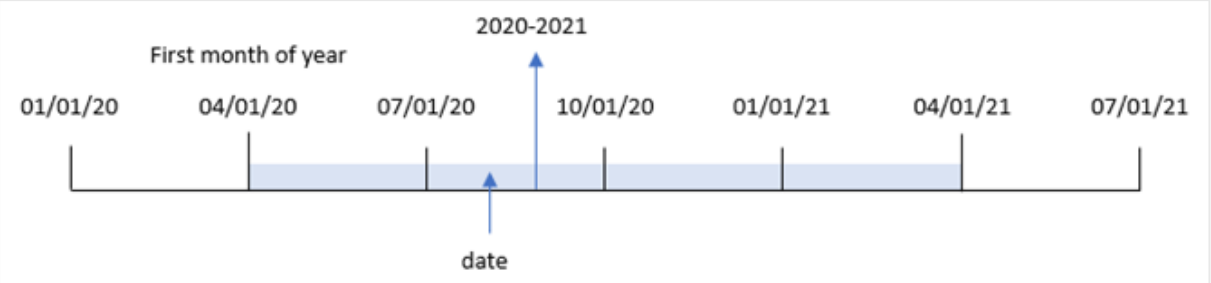
yearname() 함수의 시간 범위 다이어그램.



yearname() 함수는 평가하려는 날짜를 오프셋할 수 있고 연도의 첫 번째 달을 설정할 수 있다는 점에서 year() 함수와 다릅니다.

연도의 첫 번째 달이 1월이 아닌 경우 이 함수는 날짜가 포함된 12개월 기간에 걸쳐 두 개의 4자리 연도를 반환합니다. 예를 들어, 연도의 시작이 4월이고 평가되는 날짜가 06/30/2020인 경우 반환되는 결과는 2020-2021입니다.

4월을 연도의 첫 번째 달로 설정한 yearname() 함수 다이어그램.



구문:

```
YearName(date[, period_no[, first_month_of_year]] )
```

반환 데이터 유형: dual

인수	설명
date	평가할 날짜 또는 타임스탬프입니다.
period_no	**period_no**는 정수이며, 값 0은 **date**를 포함하는 연도를 나타냅니다. **period_no**가 음수 값일 경우 이전 연도, 양수 값일 경우 다음 연도를 나타냅니다.
first_month_of_year	1월에 시작되지 않는 (회계)연도를 사용하려는 경우 **first_month_of_year**에 2와 12 사이의 값을 지정하십시오. 그러면 표시 값이 두 연도를 보여주는 문자열이 됩니다.

다음 값을 사용하여 first_month_of_year 인수에서 연도의 첫 번째 달을 설정할 수 있습니다.

first_month_of_year 값

Month	Value
2월	2
3월	3
4월	4
5월	5
6월	6
7월	7
8월	8
9월	9
10월	10
11월	11
12월	12

사용 시기

yearname() 함수는 연도별 집계를 비교하는 데 유용합니다. 예를 들어 연도별 제품의 총 판매액을 확인하려는 경우.

이러한 차원은 마스터 캘린더 테이블에 필드를 만드는 함수를 사용하여 로드 스크립트에서 만들 수 있습니다. 차트에서 계산 차원으로 만들 수도 있습니다.

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

함수 예

예	결과
yearname('10/19/2001')	'2001'를 반환합니다.
yearname('10/19/2001',-1)	'2000'을 반환합니다.
yearname('10/19/2001',0,4)	'2001-2002'를 반환합니다.

관련 항목

항목	설명
year (page 1087)	이 함수는 포현식이 표준 숫자 해석에 따라 날짜로 해석될 경우 연도를 나타내는 정수를 반환합니다.

예 1 – 추가 인수 없음

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 'Transactions'라는 테이블에 로드되는 2020년과 2022년 사이의 트랜잭션 집합이 포함된 데이터 집합.

- 'MM/DD/YYYY'로 설정된 DateFormat 시스템 변수.
- yearname()을 사용하고 year_name 필드로 설정된 선행 LOAD.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- year_name

결과 테이블

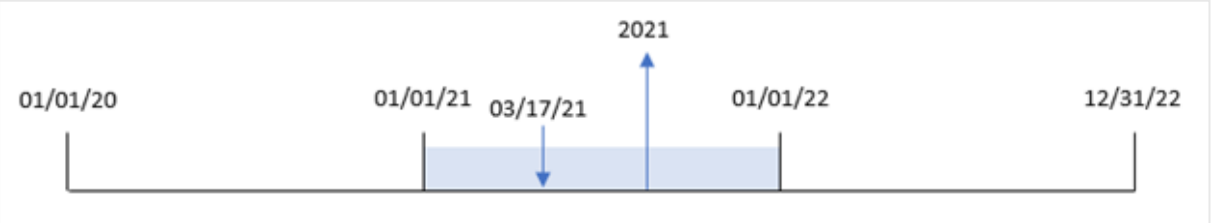
date	year_name
01/13/2020	2020

date	year_name
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

'year_name' 필드는 yearname() 함수를 사용하고 날짜 필드를 함수의 인수로 전달하여 선행 LOAD 문에서 만들어집니다.

yearname() 함수는 날짜 값이 속하는 연도를 식별하고 이를 4자리 연도 값으로 반환합니다.

2021년을 연도 값으로 표시하는 yearname() 함수 다이어그램.



예 2 – period_no

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 'Transactions'라는 테이블에 로드되는 2020년에서 2022년 사이의 트랜잭션 집합을 포함하는 데이터 집합.
- 'MM/DD/YYYY'로 설정된 DateFormat 시스템 변수.
- yearname()을 사용하고 year_name 필드로 설정된 선행 LOAD.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

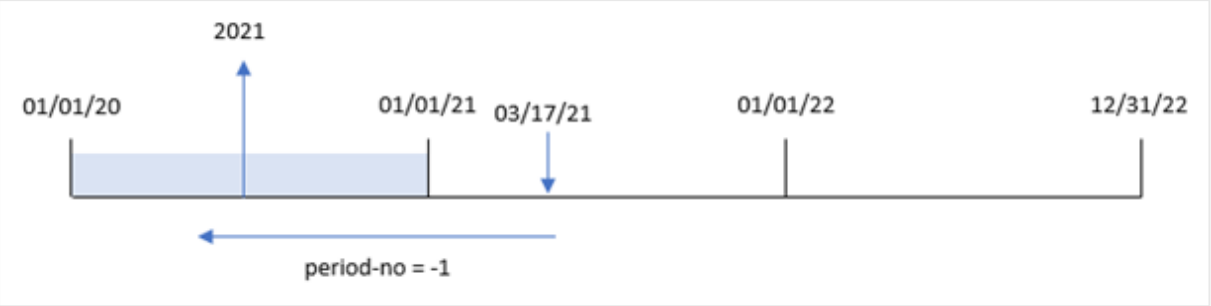
- date
- prior_year_name

결과 테이블

date	prior_year_name
01/13/2020	2019
02/26/2020	2019
03/27/2020	2019
04/16/2020	2019
05/21/2020	2019
08/14/2020	2019
10/07/2020	2019
12/05/2020	2019
01/22/2021	2020
02/03/2021	2020
03/17/2021	2020
04/23/2021	2020
05/04/2021	2020
06/30/2021	2020
07/26/2021	2020
12/27/2021	2020
06/06/2022	2021
07/18/2022	2021
11/14/2022	2021
12/12/2022	2021

period_no -1은 yearname() 함수에서 오프셋 인수로 사용되므로 함수는 먼저 트랜잭션이 발생한 연도를 식별합니다. 그런 다음 함수는 1년 전으로 이동하여 결과 연도를 반환합니다.

period_no가 -1로 설정된 yearname() 함수 다이어그램.



예 3 – first_month_of_year

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 첫 번째 예와 동일한 데이터 집합.
- 'MM/DD/YYYY'로 설정된 DateFormat 시스템 변수.
- yearname()을 사용하고 year_name 필드로 설정된 선행 LOAD.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- id
- date
- year_name

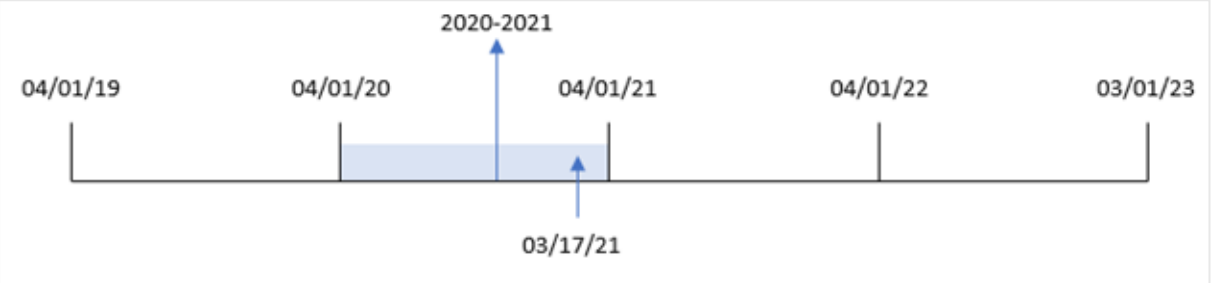
결과 테이블

date	year_name
01/13/2020	2019-2020
02/26/2020	2019-2020
03/27/2020	2019-2020
04/16/2020	2020-2021
05/21/2020	2020-2021
08/14/2020	2020-2021
10/07/2020	2020-2021
12/05/2020	2020-2021
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

yearname() 함수에서 first_month_of_year 인수 4를 사용하므로 연도의 시작은 1월 1일에서 4월 1일로 이동합니다. 따라서 각 12개월 기간은 2개의 캘린더 연도와 교차하고 yearname() 함수는 평가된 날짜에 대해 2개의 4자리 연도를 반환합니다.

트랜잭션 8198은 2021년 3월 17일에 발생합니다. yearname() 함수는 연도의 시작을 4월 1일로 설정하고 끝을 3월 30일로 설정합니다. 따라서 트랜잭션 8198은 2020년 4월 1일부터 2021년 3월 30일까지의 기간에 발생했습니다. 결과적으로 yearname() 함수는 2020-2021 값을 반환합니다.

3월을 연도의 첫 번째 달로 설정한 yearname() 함수 다이어그램.



예 4 – 차트 개체 예

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 첫 번째 예와 동일한 데이터 집합.
- 'MM/DD/YYYY'로 설정된 DateFormat 시스템 변수.

그러나 트랜잭션이 발생한 연도를 반환하는 필드는 차트 개체에서 측정값으로 만들어집니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다.

`date`
'`year_name`' 필드를 계산하려면 다음 측정값을 만듭니다.

```
=yearname(date)
```

결과 테이블

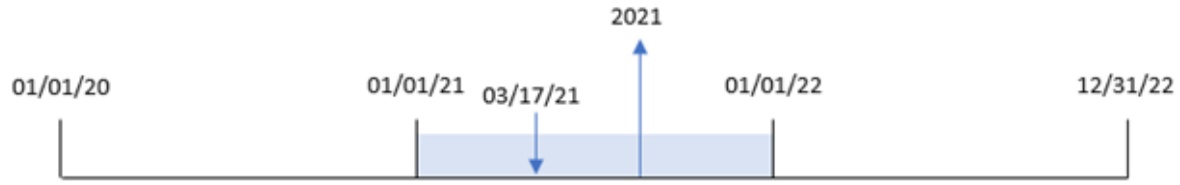
date	=yearname(date)
01/13/2020	2020
02/26/2020	2020
03/27/2020	2020
04/16/2020	2020
05/21/2020	2020
08/14/2020	2020
10/07/2020	2020
12/05/2020	2020
01/22/2021	2021
02/03/2021	2021
03/17/2021	2021
04/23/2021	2021
05/04/2021	2021
06/30/2021	2021
07/26/2021	2021
12/27/2021	2021

date	=yearname(date)
06/06/2022	2022
07/18/2022	2022
11/14/2022	2022
12/12/2022	2022

'year_name' 측정값은 yearname() 함수를 사용하고 날짜 필드를 함수의 인수로 전달하여 차트 개체에서 만들어집니다.

yearname() 함수는 날짜 값이 속하는 연도를 식별하고 이를 4자리 연도 값으로 반환합니다.

2021년을 연도 값으로 사용하는 yearname() 함수 다이어그램.



예 5 – 시나리오

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 첫 번째 예와 동일한 데이터 집합.
- 'MM/DD/YYYY'로 설정된 DateFormat 시스템 변수.

최종 사용자는 트랜잭션에 대한 분기별 총 판매를 나타내는 차트를 원합니다. 데이터 모델에서 yearname() 차원을 사용할 수 없는 경우 yearname() 함수를 계산된 차원으로 사용하여 이 차트를 만듭니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만듭니다.

연도별로 집계를 비교하려면 다음과 같이 계산된 차원을 만듭니다.

```
=yearname(date)
```

이 측정값을 만듭니다.

```
=sum(amount)
```

측정값의 **숫자 형식**을 **화폐**로 설정합니다.

결과 테이블

yearname(date)	=sum(amount)
2020	$463.55
2021	$457.69
2022	$294.35

yearstart

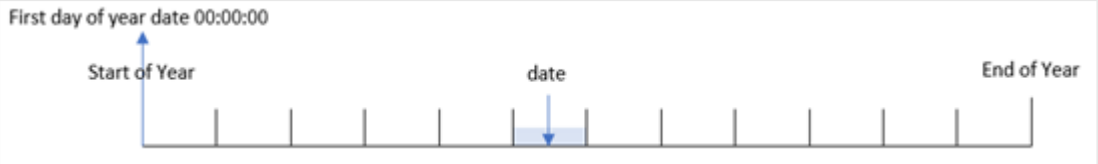
이 함수는 **date**를 포함하는 연도의 첫 번째 날의 시작에 해당하는 타임스탬프를 반환합니다. 기본 출력 형식은 스크립트에 설정된 **DateFormat**입니다.

구문:

YearStart(date[, period_no[, first_month_of_year]]**)**

즉, yearstart() 함수는 이 날짜가 속하는 연도를 확인합니다. 그런 다음 해당 연도의 첫 번째 밀리초에 대한 타임스탬프를 날짜 서식으로 반환합니다. 연도의 첫 번째 달은 기본적으로 1월입니다. 그러나 yearstart() 함수의 first_month_of_year 인수를 사용하여 첫 번째로 설정되는 월을 변경할 수 있습니다.

yearstart() 함수가 처리할 수 있는 시간 범위를 보여 주는 함수 다이어그램.



사용 시기

yearstart() 함수는 지금까지 경과된 연도의 분수를 사용하도록 계산하려는 경우 표현식의 일부로 사용됩니다. 예를 들어 연간 누계로 누적된 이자를 계산하려는 경우.

반환 데이터 유형: dual

인수

인수	설명
date	평가할 날짜 또는 타임스탬프입니다.
period_no	**period_no**는 정수이며, 값 0은 **date**를 포함하는 연도를 나타냅니다. **period_no**가 음수 값일 경우 이전 연도, 양수 값일 경우 다음 연도를 나타냅니다.
first_month_of_year	1월에 시작되지 않는 (회계)연도를 사용하려는 경우 **first_month_of_year**에 2와 12 사이의 값을 지정하십시오.

first_month_of_year argument에서 다음 달을 사용할 수 있습니다.

first_month_of_year 값

Month	Value
2월	2
3월	3
4월	4
5월	5
6월	6
7월	7
8월	8
9월	9
10월	10
11월	11
12월	12

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

함수 예

예	결과
yearstart('10/19/2001')	Returns 01/01/2001 00:00:00.
yearstart('10/19/2001',-1)	Returns 01/01/2000 00:00:00.
yearstart('10/19/2001',0,4)	Returns 04/01/2001 00:00:00.

예 1 – 기본 예

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 'Transactions'라는 테이블에 로드되는 2020년과 2022년 사이의 트랜잭션 집합이 포함된 데이터 집합.
- DateFormat 시스템 변수 서식(MM/DD/YYYY)으로 제공된 날짜 필드.
- 다음을 포함하는 선행 LOAD 문:
    - year_start 필드로 설정되는 yearstart() 함수.
    - year_start_timestamp 필드로 설정되는 Timestamp() 함수.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- id
- date
- year_start
- year_start_timestamp

결과 테이블

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

id	date	year_start	year_start_timestamp
8196	01/22/2021	01/01/2021	1/1/2021 12:00:00 AM
8197	02/03/2021	01/01/2021	1/1/2021 12:00:00 AM
8198	03/17/2021	01/01/2021	1/1/2021 12:00:00 AM
8199	04/23/2021	01/01/2021	1/1/2021 12:00:00 AM
8200	05/04/2021	01/01/2021	1/1/2021 12:00:00 AM
8201	06/30/2021	01/01/2021	1/1/2021 12:00:00 AM
8202	07/26/2021	01/01/2021	1/1/2021 12:00:00 AM
8203	12/27/2021	01/01/2021	1/1/2021 12:00:00 AM
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

'year_start' 필드는 yearstart() 함수를 사용하고 날짜 필드를 함수의 인수로 전달하여 선행 LOAD 문에서 만들어집니다.

yearstart() 함수는 날짜 값이 속하는 연도를 처음에 식별하고 해당 연도의 첫 번째 밀리초에 대한 타임스탬프를 반환합니다.

yearstart() 함수 및 트랜잭션 8199의 다이어그램.

트랜잭션 8199는 2021년 4월 23일에 발생했습니다. yearstart() 함수는 해당 연도의 첫 번째 밀리초, 즉 1월 1일 오전 12시를 반환합니다.

예 2 – period_no

로드 스크립트 및 결과

개요

첫 번째 예와 동일한 데이터 집합 및 시나리오가 사용됩니다.

그러나 이 예에서 작업은 트랜잭션이 발생한 연도 이전 연도의 시작 날짜 타임스탬프를 반환하는 필드 'previous_year_start'를 만드는 것입니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- id
- date
- previous_year_start
- previous_ year_start_timestamp

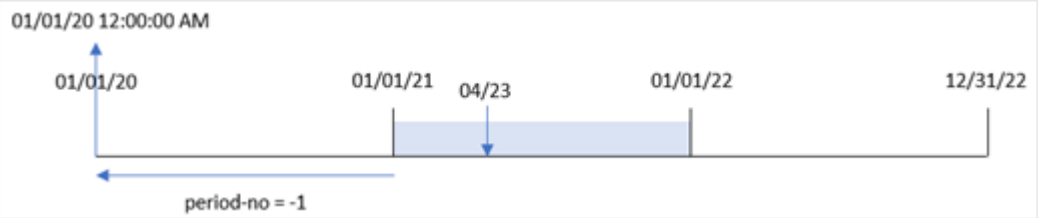
결과 테이블

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

이 경우 period_no -1은 yearstart() 함수에서 오프셋 인수로 사용되므로 함수는 먼저 트랜잭션이 발생한 연도를 식별합니다. 그런 다음 1년 전을 찾아 해당 연도의 첫 번째 밀리초를 식별합니다.

period_no가 -1인 yearstart() 함수 다이어그램.



트랜잭션 8199는 2021년 4월 23일에 발생했습니다. yearstart() 함수는 'previous_year_start' 필드에 전년도의 첫 번째 밀리초(2020년 1월 1일 오전 12시)를 반환합니다.

예 3 – first_month_of_year

로드 스크립트 및 결과

개요

첫 번째 예와 동일한 데이터 집합 및 시나리오가 사용됩니다.

그러나 이 예에서는 회사 정책이 4월 1일부터 시작되는 연도를 기준으로 합니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

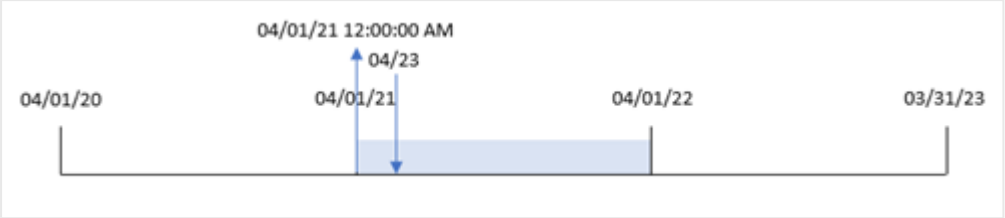
- id
- date
- year_start
- year_start_timestamp

결과 테이블

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

이 경우 yearstart() 함수에서 first_month_of_year 인수 4를 사용하므로 해당 연도의 첫 번째 날이 4월 1일로 설정되고 해당 연도의 마지막 날이 3월 31일로 설정됩니다.

4월을 첫 번째 달로 설정한 yearstart() 함수의 다이어그램.



트랜잭션 8199는 2021년 4월 23일에 발생했습니다. yearstart() 함수가 연도의 시작을 4월 1일로 설정하고 이를 트랜잭션의 'year_start' 값으로 반환하기 때문입니다.

예 4 – 차트 개체 예

로드 스크립트 및 차트 표현식

개요

첫 번째 예와 동일한 데이터 집합 및 시나리오가 사용됩니다.

그러나 이 예에서 데이터 집합은 변경되지 않고 응용 프로그램에 로드됩니다. 트랜잭션이 발생한 연도의 시작 날짜 타임스탬프를 반환하는 계산은 응용 프로그램의 차트 개체에서 측정값으로 만들어집니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- id
- date

트랜잭션이 발생한 연도를 계산하려면 다음 측정값을 만듭니다.

- =yearstart(date)
- =timestamp(yearstart(date))

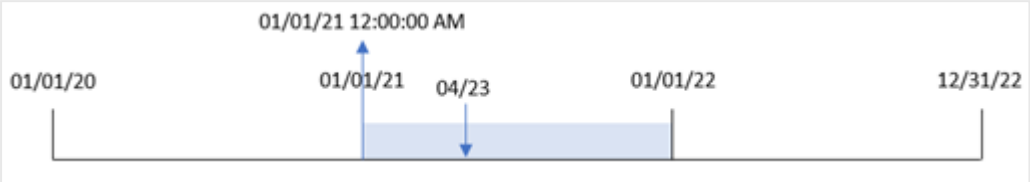
결과 테이블

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

'start_of_year' 측정값은 yearstart() 함수를 사용하고 날짜 필드를 함수의 인수로 전달하여 차트 개체에서 만들어집니다.

yearstart() 함수는 날짜 값이 속하는 연도를 처음에 식별하고 해당 연도의 첫 번째 밀리초에 대한 타임스탬프를 반환합니다.

yearstart() 함수 및 트랜잭션 8199의 다이어그램.



트랜잭션 8199는 2021년 4월 23일에 발생했습니다. yearstart() 함수는 해당 연도의 첫 번째 밀리초, 즉 1월 1일 오전 12시를 반환합니다.

예 5 – 시나리오

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 'Loans'라는 테이블에 로드되는 데이터 집합. 테이블에는 다음 필드가 포함됩니다.
  - 대출 ID.
  - 연초 잔액.
  - 연간 각 대출에 부과되는 단순 이자율.

최종 사용자는 해당 연초부터 현재까지 각 대출에 대해 발생한 현재 이자를 대출 ID별로 표시하는 차트 개체를 원합니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- loan_id
- start_balance

누적 이자를 계산하려면 다음 측정값을 만듭니다.

```
=start_balance*(rate*(today(1)-yearstart(today(1)))/365)
```

측정값의 **숫자 형식**을 **화폐**로 설정합니다.

결과 테이블

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

오늘 날짜를 유일한 인수로 사용하는 yearstart() 함수는 현재 연도의 시작 날짜를 반환합니다. 이 표현식은 현재 날짜에서 해당 결과를 빼서 올해 지금까지 경과한 일 수를 반환합니다.

그런 다음 이 값에 이자율을 곱하고 365로 나누어 해당 기간의 유효 이자율을 반환합니다. 해당 기간의 유효 이자율에 대출 시작 잔액을 곱하여 올해 지금까지 발생한 이자를 반환합니다.

yeartodate

이 함수는 스크립트를 마지막으로 로드한 날짜의 연도 내에 입력 타임스탬프가 포함되는지 파악하고, 포함되는 경우 True, 포함되지 않는 경우 False를 반환합니다.

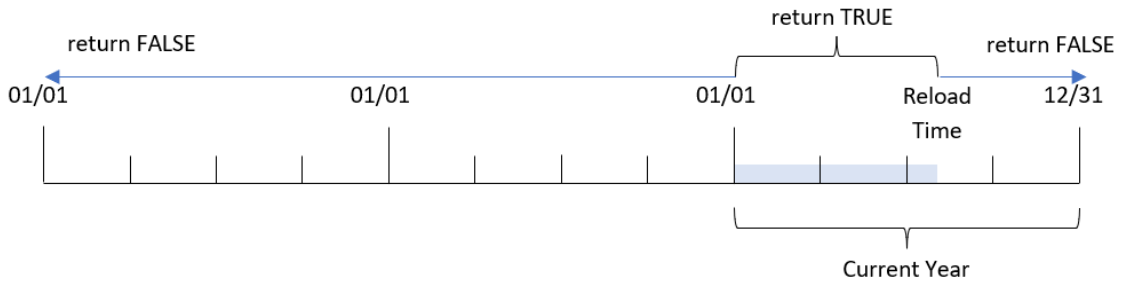
구문:

```
YearToDate(timestamp[ , yearoffset [ , firstmonth [ , todaydate] ] ])
```

반환 데이터 유형: 부울

> *Qlik Sense에서 부울 true 값은 -1로 표시되고 false 값은 0으로 표시됩니다.*

yeartodate() 함수의 다이어그램 예



옵션 매개 변수가 사용되지 않은 경우 캘린더 연도부터 현재까지는 한 해의 1월 1일에서 마지막 스크립트 실행 날짜(포함)까지의 모든 날짜를 의미합니다.

즉, 추가 매개 변수 없이 트리거될 때 yeartodate() 함수는 타임스탬프를 평가하고 다시 로드가 발생한 날짜를 포함하여 캘린더 연도 내에 날짜가 발생했는지 여부에 따라 부울 결과를 반환하는 데 사용됩니다.

그러나 firstmonth 인수를 사용하여 연도의 시작 날짜를 대체하고 yearoffset 인수를 사용하여 이전 연도 또는 다음 연도와 비교할 수도 있습니다.

마지막으로, 기록 데이터 집합의 경우 yeartodate() 함수는 todaydate를 설정하는 매개 변수를 제공합니다. 이 매개 변수는 대신 타임스탬프를 todaydate 인수에 제공된 날짜까지 포함하여 캘린더 연도와 비교합니다.

인수

인수	설명
timestamp	평가할 타임스탬프입니다(예: '10/12/2012').
yearoffset	**yearoffset**을 지정하면 다른 연도의 동일한 기간에 대해 **yeartodate**가 True를 반환합니다. 음수 **yearoffset**은 이전 연도를 나타내고 양의 오프셋은 이후의 연도를 나타냅니다. 최근 날짜는 yearoffset = -1을 지정하여 얻습니다. 생략하면 0이 사용됩니다.
firstmonth	**firstmonth**를 1과 12 사이(생략할 경우 1)로 지정하면 연도의 시작일을 원하는 달의 첫 날로 옮길 수 있습니다. 예를 들어, 5월 1일에 시작하는 회계 연도로 작업하려면 **firstmonth** = 5를 지정합니다. 값이 1이면 회계 연도가 1월 1일에 시작됨을 나타내고 값이 12이면 12월 1일부터 시작하는 회계 연도를 나타냅니다.
todaydate	**todaydate**(생략할 경우 마지막 스크립트 실행 시의 타임스탬프)를 지정하면 기간의 상한으로 사용된 날짜를 이동할 수 있습니다.

사용 시기

yeartodate() 함수는 부울 결과를 반환합니다. 일반적으로 이 유형의 함수는 if 표현식의 조건으로 사용됩니다. 이렇게 하면 평가 날짜가 응용 프로그램의 마지막 다시 로드 날짜를 포함하여 해당 연도에 발생했는지 여부에 따라 집계 또는 계산이 반환됩니다.

예를 들어 YearToDate() 함수를 사용하여 현재 연도에서 지금까지 제조된 모든 장비를 식별할 수 있습니다.

다음 예에서는 마지막 다시 로드 시간 = 11/18/2011을 가정합니다.

함수 예

예	결과
`yeartodate( '11/18/2010')`	반환 값: False
`yeartodate( '02/01/2011')`	반환 값: True
`yeartodate( '11/18/2011')`	반환 값: True
`yeartodate( '11/19/2011')`	반환 값: False
`yeartodate( '11/19/2011', 0, 1, '12/31/2011')`	반환 값: True
`yeartodate( '11/18/2010', -1)`	반환 값: True

예	결과
yeartodate('11/18/2011', -1)	반환 값: False
yeartodate('04/30/2011', 0, 5)	반환 값: False
yeartodate('05/01/2011', 0, 5)	반환 값: True

국가별 설정

달리 지정하지 않는 한 이 항목의 예에서는 다음 날짜 형식을 사용합니다. YYYY/MM/DD. 날짜 형식은 데이터 로드 스크립트의 SET DateFormat 문에 지정됩니다. 기본 날짜 형식은 지역 설정 및 기타 요인으로 인해 시스템에서 다를 수 있습니다. 아래 예의 형식을 요구 사항에 맞게 변경할 수 있습니다. 또는 이러한 예와 일치하도록 로드 스크립트의 형식을 변경할 수 있습니다.

앱의 기본 국가별 설정은 Qlik Sense가 설치된 컴퓨터 또는 서버의 국가별 시스템 설정을 기반으로 합니다. 액세스하는 Qlik Sense 서버가 스웨덴으로 설정된 경우 데이터 로드 편집기는 날짜, 시간 및 통화에 대해 스웨덴 지역 설정을 사용합니다. 이러한 국가별 형식 설정은 Qlik Sense 사용자 인터페이스에 표시되는 언어와 관련이 없습니다. Qlik Sense는 사용 중인 브라우저와 동일한 언어로 표시됩니다.

예 1 – 기본 예

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Transactions라는 테이블에 로드되는 2020년과 2022년 사이의 트랜잭션 집합이 포함된 데이터 집합.
- DateFormat 시스템 변수(MM/DD/YYYY) 서식으로 제공된 날짜 필드.
- 마지막 다시 로드 날짜까지 캘린더 연도에 발생한 트랜잭션을 확인하는 필드 year_to_date 만들기.

작성 시 날짜는 2022년 4월 26일입니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

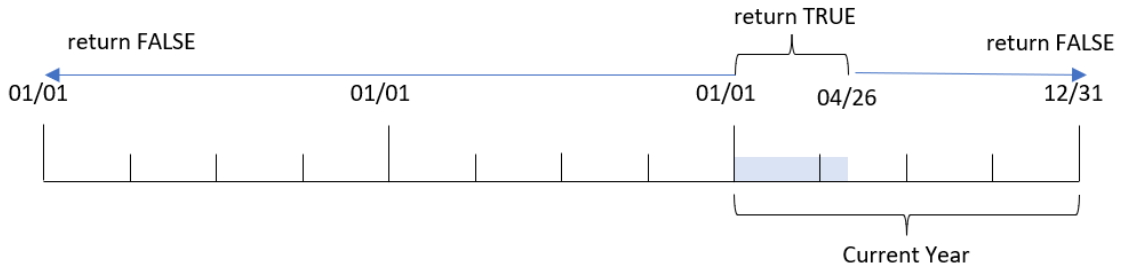
- `date`
- `year_to_date`

결과 테이블

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

date	year_to_date
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

yeartodate() 함수의 다이어그램, 기본 예



year_to_date 필드는 yeartodate() 함수를 사용하고 date 필드를 함수의 인수로 전달하여 선행 LOAD 문에서 만들어집니다.

더 이상의 매개 변수가 함수에 전달되지 않기 때문에 yeartodate() 함수는 초기에 다시 로드 날짜를 식별하므로 부울 결과 TRUE를 반환하는 현재 캘린더 연도(1월 1일부터 시작)의 경계를 식별합니다.

따라서 다시 로드 날짜인 1월 1일과 4월 26일 사이에 발생하는 모든 트랜잭션은 부울 결과 TRUE를 반환합니다. 2022년이 시작되기 전에 발생한 모든 트랜잭션은 부울 결과 FALSE를 반환합니다.

예 2 – yearoffset

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 첫 번째 예와 동일한 데이터 집합 및 시나리오.
- 캘린더 연도부터 현재까지 2년 전 발생한 트랜잭션을 확인하는 필드 two_years_prior 만들기.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- two_years_prior

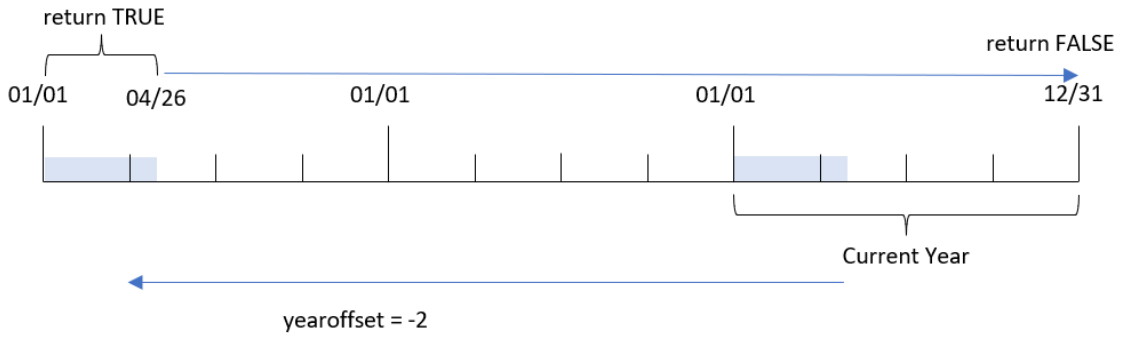
결과 테이블

date	two_years_prior
01/10/2020	-1
02/28/2020	-1
04/09/2020	-1
04/16/2020	-1

date	two_years_prior
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

yeartodate() 함수의 yearoffset 인수로 -2를 사용하여 함수는 비교 캘린더 연도 세그먼트의 경계를 2년 전체만큼 시프트합니다. 초기에 연도 세그먼트는 2022년 1월 1일에서 4월 26일 사이에 해당합니다. 그런 다음 yearoffset 인수는 이 세그먼트를 2년 전으로 오프셋합니다. 날짜 경계는 2020년 1월 1일에서 4월 26일 사이가 됩니다.

yeartodate() 함수의 다이어그램, yearoffset 예



따라서 2020년 1월 1일과 4월 26일 사이에 발생하는 모든 트랜잭션은 부울 결과 TRUE를 반환합니다. 이 세그먼트 이전 또는 이후에 나타나는 모든 트랜잭션이 FALSE를 반환합니다.

예 3 – firstmonth

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 첫 번째 예와 동일한 데이터 집합 및 시나리오.
- 마지막 다시 로드 날짜까지 캘린더 연도에 발생한 트랜잭션을 확인하는 필드 year_to_date 만들기.

이 예에서는 회계 연도의 시작을 7월 1일로 설정합니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

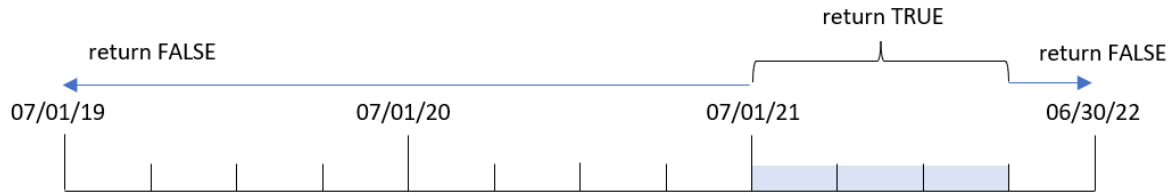
- date
- year_to_date

결과 테이블

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	-1
12/27/2021	-1
02/02/2022	-1
02/26/2022	-1
03/07/2022	-1
03/11/2022	-1

이 경우 yeartodate() 함수에서 firstmonth 인수 7을 사용하므로 해당 연도의 첫 번째 날이 7월 1일로 설정되고 해당 연도의 마지막 날이 6월 30일로 설정됩니다.

yeartodate() 함수 다이어그램, firstmonth 예



따라서 2021년 7월 1일과 다시 로드 날짜인 2022년 4월 26일 사이에 발생한 모든 트랜잭션은 부울 결과 TRUE를 반환합니다. 2021년 7월 1일 이전에 발생한 모든 트랜잭션은 부울 결과 FALSE를 반환합니다.

예 4 – todaydate

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- 첫 번째 예와 동일한 데이터 집합 및 시나리오.
- 마지막 다시 로드 날짜까지 캘린더 연도에 발생한 트랜잭션을 확인하는 필드 year_to_date 만들기.

그러나 이 예에서는 2022년 3월 1일을 포함하여 캘린더 연도에 발생한 모든 트랜잭션을 식별해야 합니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- date
- year_to_date

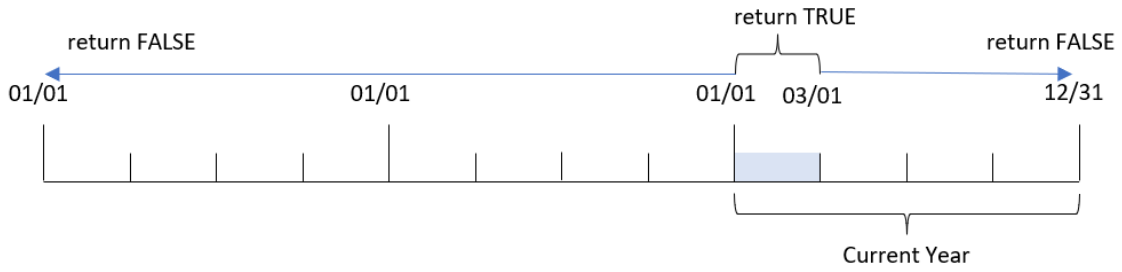
결과 테이블

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

date	year_to_date
02/26/2022	-1
03/07/2022	0
03/11/2022	0

이 경우 todaydate 인수 03/01/2022가 yeartodate() 함수에서 사용되므로 비교 캘린더 연도 세그먼트의 끝 경계를 2022년 3월 1일로 설정합니다. firstmonth 매개 변수(1 ~ 12)를 제공하는 것이 중요합니다. 그렇지 않으면 함수는 null 결과를 반환합니다.

yeartodate() 함수의 다이어그램, todaydate 인수를 사용한 예



따라서 todaydate 매개 변수인 2022년 1월 1일과 2022년 3월 1일 사이에 발생하는 모든 트랜잭션은 부울 결과 TRUE를 반환합니다. 2022년 1월 1일 이전 또는 2022년 3월 1일 이후에 발생한 모든 트랜잭션은 부울 결과 FALSE를 반환합니다.

예 5 – 차트 개체 예

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

이 로드 스크립트에는 첫 번째 예와 동일한 데이터 집합 및 시나리오가 포함되어 있습니다.

그러나 이 예에서는 변경되지 않은 데이터 집합이 응용 프로그램에 로드됩니다. 마지막 다시 로드 날짜까지 캘린더 연도에 발생한 트랜잭션을 확인하는 계산은 응용 프로그램의 차트 개체에서 측정값으로 만들어집니다.

로드 스크립트

```
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
];
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 이 필드를 차원으로 추가합니다. date.

다음 측정값을 추가합니다.

```
=yeartodate(date)
```

결과 테이블

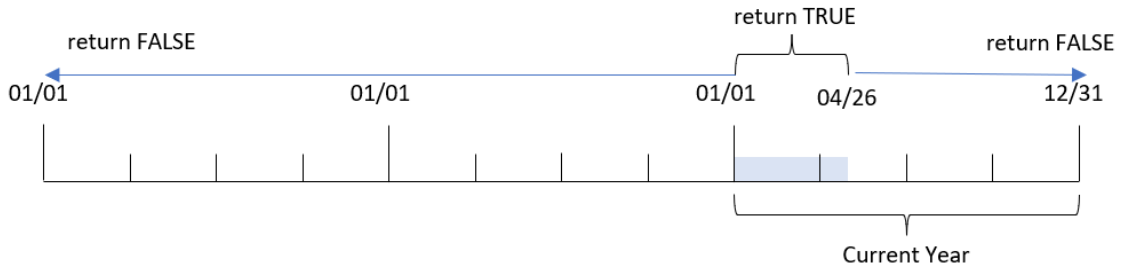
date	=yeartodate(date)
01/10/2020	0
02/28/2020	0
04/09/2020	0
04/16/2020	0
05/21/2020	0
08/14/2020	0
10/07/2020	0
12/05/2020	0
01/22/2021	0
02/03/2021	0
03/17/2021	0
04/23/2021	0

date	=yeartodate(date)
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

year_to_date 측정값은 yeartodate() 함수를 사용하고 date 필드를 함수의 인수로 전달하여 차트 개체에서 만들어집니다.

더 이상의 매개 변수가 함수에 전달되지 않기 때문에 yeartodate() 함수는 초기에 다시 로드 날짜를 식별하므로 부울 결과 TRUE를 반환하는 현재 캘린더 연도(1월 1일부터 시작)의 경계를 식별합니다.

yeartodate() 함수의 다이어그램, 차트 개체를 사용한 예



다시 로드 날짜인 1월 1일과 4월 26일 사이에 발생하는 모든 트랜잭션은 부울 결과 TRUE를 반환합니다. 2022년이 시작되기 전에 발생한 모든 트랜잭션은 부울 결과 FALSE를 반환합니다.

예 6 – 시나리오

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- Transactions라는 테이블에 로드되는 2020년과 2022년 사이의 트랜잭션 집합이 포함된 데이터 집합.
- DateFormat 시스템 변수(MM/DD/YYYY) 서식으로 제공된 날짜 필드.

최종 사용자는 2021년 해당 기간 동안의 총 판매를 마지막 다시 로드 시간과 같은 현재 연도부터 현재까지로 표시하는 KPI 개체를 원합니다.

작성 시 날짜는 2022년 6월 16일입니다.

로드 스크립트

```
SET DateFormat='MM/DD/YYYY';

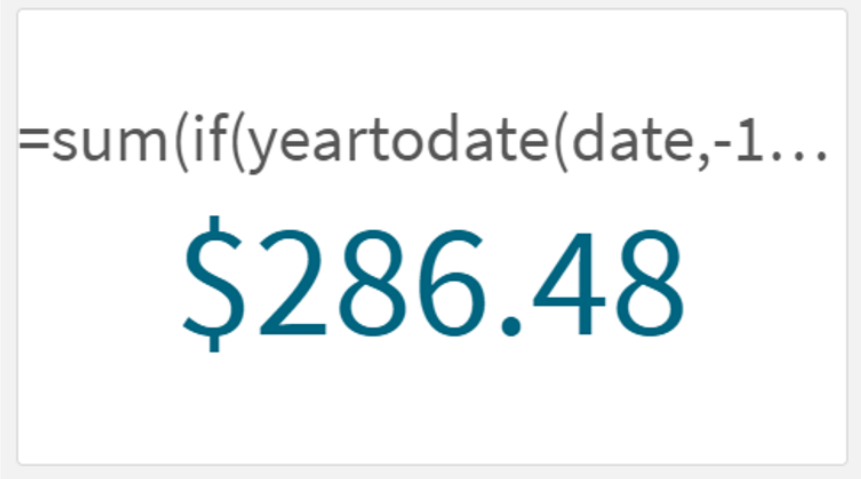
Transactions:
Load
*
Inline
[
id,date,amount
8188,01/10/2020,37.23
8189,02/28/2020,17.17
8190,04/09/2020,88.27
8191,04/16/2020,57.42
8192,05/21/2020,53.80
8193,08/14/2020,82.06
8194,10/07/2020,40.39
8195,12/05/2020,87.21
8196,01/22/2021,95.93
8197,02/03/2021,45.89
8198,03/17/2021,36.23
8199,04/23/2021,25.66
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
];
```

결과

다음과 같이 하십시오.

1. KPI 개체를 만듭니다.
2. 총 판매액을 계산하려면 다음 집계 측정값을 만듭니다.
   `=sum(if(yeartodate(date,-1),amount,0))`
3. 측정값의 **숫자 형식**을 **화폐**로 설정합니다.

2021년 KPI yeartodate() 차트



yeartodate() 함수는 각 트랜잭션 ID의 날짜를 평가할 때 부울 값을 반환합니다. 다시 로드가 2022년 6월 16일에 발생했으므로 yeartodate 함수는 연도 기간을 2022년 1월 1일에서 2022년 6월 16일 사이로 분할합니다. 그러나 period_no 값 -1이 이 함수에 사용되었으므로 이러한 경계는 이전 연도로 시프트됩니다. 따라서 2022년 1월 1일에서 2022년 6월 16일 사이에 발생하는 모든 트랜잭션에 대해 yeartodate() 함수는 부울 값 TRUE를 반환하고 금액을 합산합니다.

8.8 지수 및 로그 함수

이 섹션에서는 지수 및 로그 계산과 관련된 함수를 설명합니다. 모든 함수는 데이터 로드 스크립트와 차트 표현식 모두에서 사용할 수 있습니다.

아래 함수에서 매개 변수는 **x** 및 **y**를 실수로 해석해야 하는 표현식입니다.

exp
자연 로그 **e**를 밑으로 사용하는 자연 지수 함수 e^x입니다. 결과는 양수입니다.

```
exp(x )
```

예 및 결과:

exp(3)은 20.085를 반환합니다.

log
x의 자연 로그입니다. 이 함수는 **x**> 0인 경우에만 정의됩니다. 결과는 숫자입니다.

```
log(x )
```

예 및 결과:

log(3)은 1.0986을 반환합니다.

log10
10을 밑으로 하는 **x**의 상용 로그입니다. 이 함수는 **x**> 0인 경우에만 정의됩니다. 결과는 숫자입니다.

```
log10(x )
```

예 및 결과:

log10(3)은 0.4771을 반환합니다.

pow
x의 **y** 승을 반환합니다. 결과는 숫자입니다.

```
pow(x,y )
```

예 및 결과:

pow(3, 3)은 27을 반환합니다.

sqr
x 제곱(**x**의 2승)입니다. 결과는 숫자입니다.

```
sqr (x )
```

예 및 결과:

sqr(3)은 9를 반환합니다.

sqrt
x의 제곱근입니다. 이 함수는 **x** >= 0인 경우에만 정의됩니다. 결과는 양수입니다.

```
sqrt(x )
```

예 및 결과:

sqrt(3)은 1.732를 반환합니다.

8.9 필드 함수

이 함수는 차트 표현식에서만 사용할 수 있습니다.

필드 함수는 필드 선택 내용의 다양한 측면을 식별하는 정수 또는 문자열 중 하나를 반환합니다.

카운트 함수

GetAlternativeCount
GetAlternativeCount()는 식별된 필드에서 대체 가능(연한 회색) 값의 수를 찾는 데 사용됩니다.

```
GetAlternativeCount - 차트 함수 (field_name)
```

GetExcludedCount

GetExcludedCount() 식별된 필드에서 제외된 고유 값의 수를 찾습니다. 제외된 값에는 대체(옅은 회색), 제외(짙은 회색) 및 선택된 제외(체크 표시가 있는 짙은 회색) 필드가 포함됩니다.

```
GetExcludedCount - 차트 함수 (page 1149)(field_name)
```

GetNotSelectedCount

이 차트 함수는 이름이 **fieldname**인 필드 내에서 선택되지 않은 값의 수를 반환합니다. 이 함수가 연관성을 가지려면 필드가 AND 모드여야 합니다.

```
GetNotSelectedCount - 차트 함수(fieldname [, includeexcluded=false])
```

GetPossibleCount

GetPossibleCount()는 식별된 필드에서 사용 가능한 값의 수를 찾는 데 사용됩니다. 식별된 필드에 선택 내용이 포함되어 있으면 선택한(녹색) 필드를 카운트합니다. 그렇지 않으면 연결된(흰색) 값을 카운트합니다.

```
GetPossibleCount - 차트 함수(field_name)
```

GetSelectedCount

GetSelectedCount()는 필드에서 선택한(녹색) 값의 수를 찾습니다.

```
GetSelectedCount - 차트 함수 (field_name [, include_excluded])
```

필드 및 선택 함수

GetCurrentSelections

GetCurrentSelections()는 앱에서 현재 선택 목록을 반환합니다. 검색 상자에서 검색 문자열을 사용하여 선택된 경우 **GetCurrentSelections()**는 검색 문자열을 반환합니다.

```
GetCurrentSelections - 차트 함수([record_sep [,tag_sep [,value_sep [,max_
values]]]])
```

GetFieldSelections

GetFieldSelections()는 필드 내에서 현재 선택 내용이 있는 **문자열**을 반환합니다.

```
GetFieldSelections - 차트 함수 ( field_name [, value_sep [, max_values]])
```

GetObjectDimension

GetObjectDimension()은 차원의 이름을 반환합니다. **Index**는 어떤 차원을 반환해야 하는지 지정하는 선택적 정수입니다.

```
GetObjectDimension - 차트 함수 ([index])
```

GetObjectField

GetObjectField()는 차원의 이름을 반환합니다. **Index**는 어떤 차원을 반환해야 하는지 지정하는 선택적 정수입니다.

```
GetObjectField - 차트 함수 ([index])
```

GetObjectMeasure

GetObjectMeasure()는 차원의 이름을 반환합니다. **Index**는 어떤 측정값을 반환해야 하는지 지정하는 선택적 정수입니다.

```
GetObjectMeasure - 차트 함수 ([index])
```

GetAlternativeCount - 차트 함수

GetAlternativeCount()는 식별된 필드에서 대체 가능(연한 회색) 값의 수를 찾는 데 사용됩니다.

구문:

```
GetAlternativeCount (field_name)
```

반환 데이터 유형: 정수

인수:

인수

인수	설명
field_name	측정할 데이터 범위가 포함된 필드입니다.

예 및 결과:

다음 예에서는 필터 창에 로드된 **First name** 필드를 사용합니다.

예 및 결과

예	결과
First name에서 **John**을 선택한 것으로 가정합니다. `GetAlternativeCount ([First name])`	4이며, **First name**에 고유하며 제외된(회색) 값이 4개이기 때문입니다.
John 및 **Peter**를 선택한 것으로 가정합니다. `GetAlternativeCount ([First name])`	3이며, **First name**에 고유하며 제외된(회색) 값이 3개이기 때문입니다.
First name에서 아무 값도 선택하지 않은 것으로 가정합니다. `GetAlternativeCount ([First name])`	0이며, 선택 내용이 없기 때문입니다.

데이터 사용 예:

```
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 - 차트 함수

GetCurrentSelections()는 앱에서 현재 선택 목록을 반환합니다. 검색 상자에서 검색 문자열을 사용하여 선택된 경우 **GetCurrentSelections()**는 검색 문자열을 반환합니다.

옵션을 사용할 경우 record_sep을 지정해야 합니다. 새로운 줄을 지정하려면 **record_sep**을 **chr(13)&chr(10)**으로 설정합니다.

둘을 제외한 모두 또는 하나를 제외한 모든 값을 선택한 경우 각각 'NOT x,y' 또는 'NOT y' 형식이 사용됩니다. 모든 값을 선택하고 모든 값의 카운트가 max_values보다 클 경우 텍스트 ALL이 반환됩니다.

구문:

```
GetCurrentSelections ([record_sep [, tag_sep [, value_sep [, max_values [, state_name]]]]])
```

반환 데이터 유형: 문자열

인수:

인수

인수	설명
record_sep	필드 레코드 사이에 입력할 구분 기호입니다. 기본값은 새 줄을 의미하는 <CR><LF>입니다.
tag_sep	필드 이름 태그와 필드 값 사이에 입력할 구분 기호입니다. 기본값은 ': '입니다.
value_sep	필드 값 사이에 입력할 구분 기호입니다. 필드 값 사이에 입력할 구분 기호입니다. 기본값은 ', '입니다.
max_values	개별적으로 나열할 필드 값의 최대 수입니다. 이보다 많은 값을 선택하면 'x of y 값' 형식이 대신 사용됩니다. 기본값은 6입니다.
state_name	특정 시각화에 대해 선택한 대체 상태의 이름입니다. **state_name** 인수를 사용하는 경우 지정된 상태 이름과 연관된 선택 내용만 고려됩니다.

예 및 결과:

다음 예에서는 서로 다른 필터 창에 로드된 두 필드를 사용하며, 하나는 **First name** 이름, 하나는 **Initials**입니다.

예 및 결과

예	결과
First name에서 **John**을 선택한 것으로 가정합니다. `GetCurrentSelections ()`	'First name: John'

예	결과
First name에서 **John** 및 **Peter**를 선택한 것으로 가정합니다. `GetCurrentSelections ()`	'First name: John, Peter'
First name에서 **John** 및 **Peter**를 선택하고 **Initials**에서 **JA**를 선택한 것으로 가정합니다. `GetCurrentSelections ()`	'First name: John, Peter Initials: JA'
First name에서 **John**을 선택하고 **Initials**에서 **JA**를 선택한 것으로 가정합니다. `GetCurrentSelections ( chr(13)&chr(10) , ' = ' )`	'First name = John Initials = JA'
First name에서 Sue를 제외한 모든 이름을 선택하고 **Initials**에서 아무 것도 선택하지 않은 것으로 가정합니다. `GetCurrentSelections (chr(13)&chr(10), '=', ',' ,3)`	'First name=NOT Sue'

데이터 사용 예:

```
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 - 차트 함수

GetExcludedCount() 식별된 필드에서 제외된 고유 값의 수를 찾습니다. 제외된 값에는 대체(옅은 회색), 제외(짙은 회색) 및 선택된 제외(체크 표시가 있는 짙은 회색) 필드가 포함됩니다.

구문:

```
GetExcludedCount (field_name)
```

반환 데이터 유형: 문자열

인수:

인수

인수	설명
field_name	측정할 데이터 범위가 포함된 필드입니다.

예 및 결과:

다음 예에서는 서로 다른 필터 창에 로드된 세 필드를 사용하며, 하나는 **First name**, 하나는 **Last name**, 하나는 **Initials**입니다.

예 및 결과

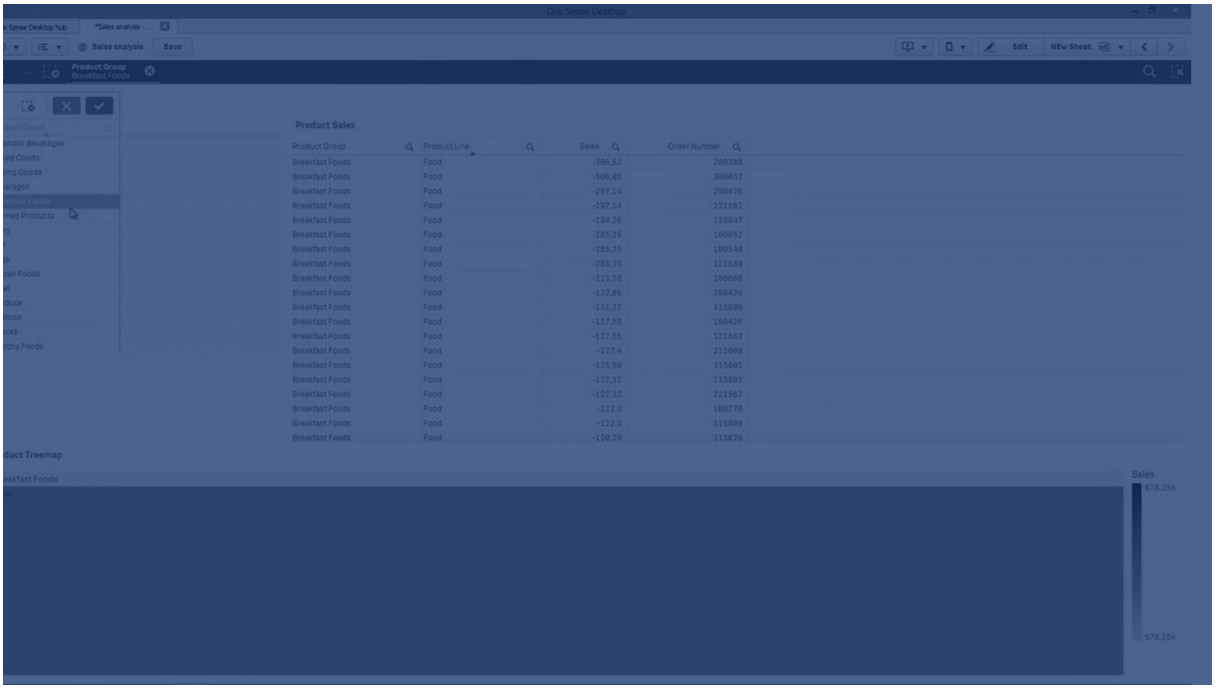
예	결과
First name에서 아무 값도 선택되지 않은 경우	`GetExcludedCount (Initials) = 0` 선택 내용이 없습니다.
First name에서 **John**이 선택된 경우	`GetExcludedCount (Initials) = 5` **Initials**에 제외된 값이 5개입니다(짙은 회색). 여섯 번째 셀(JA)은 흰색이며, **First name**의 선택 내용 John과 연결되었기 때문입니다.
John과 **Peter**가 선택된 경우	`GetExcludedCount (Initials) = 3` **Initials**에서 John은 1개의 값과 연결되어 있고 Peter는 2개의 값과 연결되어 있습니다.
John 및 **Peter**가 **First name**에서 선택된 경우 **Franc**가 **Last name**에서 선택됩니다.	`GetExcludedCount ([First name]) = 4` **First name**에 제외된 값이 4개입니다(짙은 회색). **GetExcludedCount()**는 대체 및 선택된 제외 필드를 포함하여 제외된 값을 가진 필드에 대해 평가합니다.
John 및 **Peter**가 **First name**에서 선택된 경우 **Franc** 및 **Anderson**이 **Last name**에서 선택됩니다.	`GetExcludedCount (Initials) = 4` **Initials**에 제외된 값이 4개입니다(짙은 회색). 다른 두 개의 셀(JA 및 PF)은 흰색이며 **First name**의 선택 내용 John 및 Peter가 연결되었기 때문입니다.
John 및 **Peter**가 **First name**에서 선택된 경우 **Franc** 및 **Anderson**이 **Last name**에서 선택됩니다.	`GetExcludedCount ([Last name]) = 4` **Initials**에 제외된 값이 4개입니다. Devonshire이 옅은 회색인 반면 Brown, Carr, Elliot은 짙은 회색입니다.

데이터 사용 예:

```
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 - 차트 함수

GetFieldSelections()는 필드 내에서 현재 선택 내용이 있는 **문자열**을 반환합니다.



둘을 제외한 모두 또는 하나를 제외한 모든 값을 선택한 경우 각각 'NOT x,y' 또는 'NOT y' 형식이 사용됩니다. 모든 값을 선택하고 모든 값의 카운트가 max_values보다 클 경우 텍스트 ALL이 반환됩니다.

구문:

```
GetFieldSelections ( field_name [, value_sep [, max_values [, state_name]]])
```

반환 데이터 유형: 문자열

문자열 형식 반환

서식	설명
'a, b, c'	선택한 값의 수가 max_values 이하인 경우 반환되는 문자열은 선택한 값의 목록입니다. 이러한 값은 value_sep를 구분 기호로 사용하여 구분됩니다.
'NOT a, b, c'	선택되지 않은 값의 수가 max_values 이하이면 반환되는 문자열은 NOT을 접두사로 사용하는 선택되지 않은 값의 목록입니다. 이러한 값은 value_sep를 구분 기호로 사용하여 구분됩니다.
'x of y'	x = 선택한 값 수 y = 총 값 수 이는 max_values < x < (y - max_values)인 경우 반환됩니다.
'ALL'	모든 값이 선택되면 반환됩니다.
'-'	값이 선택되지 않은 경우 반환됩니다.
<search string>	검색을 사용하여 선택한 경우 검색 문자열이 반환됩니다.

인수:

인수

인수	설명
field_name	측정할 데이터 범위가 포함된 필드입니다.
value_sep	필드 값 사이에 입력할 구분 기호입니다. 필드 값 사이에 입력할 구분 기호입니다. 기본값은 ', '입니다.
max_values	개별적으로 나열할 필드 값의 최대 수입니다. 이보다 많은 값을 선택하면 'x of y 값' 형식이 대신 사용됩니다. 기본값은 6입니다.
state_name	특정 시각화에 대해 선택한 대체 상태의 이름입니다. **state_name** 인수를 사용하는 경우 지정된 상태 이름과 연관된 선택 내용만 고려됩니다.

예 및 결과:

다음 예에서는 필터 창에 로드된 **First name** 필드를 사용합니다.

예 및 결과

예	결과
First name에서 **John**을 선택한 것으로 가정합니다. `GetFieldSelections ([First name])`	'John'
John 및 **Peter**를 선택한 것으로 가정합니다. `GetFieldSelections ([First name])`	'John,Peter'
John 및 **Peter**를 선택한 것으로 가정합니다. `GetFieldSelections ([First name],'; ')`	'John; Peter'
First name에서 **John**, **Sue**, **Mark**를 선택한 것으로 가정합니다. `GetFieldSelections ([First name],';',2)`	'NOT Jane;Peter'이며, 값 2가 max_values 인수의 값으로 선언되었기 때문입니다. 그렇지 않으면 결과는 John; Sue; Mark.가 되었을 것입니다.

데이터 사용 예:

```
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 - 차트 함수

이 차트 함수는 이름이 **fieldname**인 필드 내에서 선택되지 않은 값의 수를 반환합니다. 이 함수가 연관성을 가지려면 필드가 AND 모드여야 합니다.

구문:

```
GetNotSelectedCount(fieldname [, includeexcluded=false])
```

인수:

인수

인수	설명
fieldname	평가할 필드의 이름입니다.
includeexcluded	**includeexcluded**가 True로 지정된 경우 카운트에 다른 필드의 선택 내용에 따라 제외된 선택 값이 포함됩니다.

```
GetNotSelectedCount( Country )
```

```
GetNotSelectedCount( Country, true )
```

GetObjectDimension - 차트 함수

GetObjectDimension()은 차원의 이름을 반환합니다. **Index**는 어떤 차원을 반환해야 하는지 지정하는 선택적 정수입니다.

차트의 제목, 부제, 바닥글, 참조선 표현식 및 최소/최대 표현식 위치에서는 이 함수를 사용할 수 없습니다.

Object ID를 사용하여 다른 개체에서 차원 또는 측정값의 이름을 참조할 수 없습니다.

구문:

```
GetObjectDimension ([index])
```

```
GetObjectDimension(1)
```

예: 차트 표현식
차트 표현식에서 GetObjectDimension 함수의 예를 보여 주는 Qlik Sense 테이블

transaction_date	customer_id	transaction_quantity	=GetObjectDimension ()	=GetObjectDimension (0)	=GetObjectDimension (1)
2018/08/30	049681	13	transaction_date	transaction_date	customer_id
2018/08/30	203521	6	transaction_date	transaction_date	customer_id
2018/08/30	203521	21	transaction_date	transaction_date	customer_id

측정값의 이름을 반환하려면 대신 **GetObjectMeasure** 함수를 사용하십시오.

GetObjectField - 차트 함수

GetObjectField()는 차원의 이름을 반환합니다. **Index**는 어떤 차원을 반환해야 하는지 지정하는 선택적 정수입니다.

차트의 제목, 부제, 바닥글, 참조선 표현식 및 최소/최대 표현식 위치에서는 이 함수를 사용할 수 없습니다.

Object ID를 사용하여 다른 개체에서 차원 또는 측정값의 이름을 참조할 수 없습니다.

구문:

```
GetObjectField ([index])
```

```
GetObjectField(1)
```

예: 차트 표현식
차트 표현식에서 GetObjectField 함수의 예를 보여 주는 Qlik Sense 테이블입니다.

transaction_date	customer_id	transaction_quantity	GetObjectField	=GetObjectField (0)	=GetObjectField (1)
2018/08/30	049681	13	transaction_date	transaction_date	customer_id
2018/08/30	203521	6	transaction_date	transaction_date	customer_id
2018/08/30	203521	21	transaction_date	transaction_date	customer_id

측정값의 이름을 반환하려면 대신 **GetObjectMeasure** 함수를 사용하십시오.

GetObjectMeasure - 차트 함수

GetObjectMeasure()는 차원의 이름을 반환합니다. **Index**는 어떤 측정값을 반환해야 하는지 지정하는 선택적 정수입니다.

차트의 제목, 부제, 바닥글, 참조선 표현식 및 최소/최대 표현식 위치에서는 이 함수를 사용할 수 없습니다.

Object ID를 사용하여 다른 개체에서 차원 또는 측정값의 이름을 참조할 수 없습니다.

구문:

```
GetObjectMeasure ([index])
```

```
GetObjectMeasure(1)
```

예: 차트 표현식

차트 표현식에서 GetObjectMeasure 함수의 예를 보여 주는 Qlik Sense 테이블

customer_id	sum (transaction_quantity)	Avg (transaction_quantity)	=GetObjectMeasure ()	=GetObjectMeasure(0)	=GetObjectMeasure(1)
49681	13	13	sum(transaction_quantity)	sum(transaction_quantity)	Avg(transaction_quantity)
203521	27	13.5	sum(transaction_quantity)	sum(transaction_quantity)	Avg(transaction_quantity)

차원 이름을 반환하려면 대신 **GetObjectField** 함수를 사용하십시오.

GetPossibleCount - 차트 함수

GetPossibleCount()는 식별된 필드에서 사용 가능한 값의 수를 찾는 데 사용됩니다. 식별된 필드에 선택 내용이 포함되어 있으면 선택한(녹색) 필드를 카운트합니다. 그렇지 않으면 연결된(흰색) 값을 카운트합니다.

선택 내용이 있는 필드의 경우, **GetPossibleCount()**에서 선택된(녹색) 필드의 수를 반환합니다.

반환 데이터 유형: 정수

구문:

```
GetPossibleCount (field_name)
```

인수:

인수

인수	설명
field_name	측정할 데이터 범위가 포함된 필드입니다.

예 및 결과:

다음 예에서는 서로 다른 필터 창에 로드된 두 필드를 사용하며, 하나는 **First name** 이름, 하나는 **Initials**입니다.

예 및 결과

예	결과
First name에서 **John**을 선택한 것으로 가정합니다. `GetPossibleCount ([Initials])`	1이며, **First name**에서 선택한 **John**과 연결된 이니셜에 하나의 값이 있기 때문입니다.
First name에서 **John**을 선택한 것으로 가정합니다. `GetPossibleCount ([First name])`	1이며, **First name**에 하나의 선택 내용, **John**이 있기 때문입니다.
First name에서 **Peter**를 선택한 것으로 가정합니다. `GetPossibleCount ([Initials])`	2이며, Peter가 **Initials**의 값 2개와 연결되었기 때문입니다.
First name에서 아무 값도 선택하지 않은 것으로 가정합니다. `GetPossibleCount ([First name])`	5이며, 선택 내용이 없고 **First name**에 고유한 값이 5개이기 때문입니다.
First name에서 아무 값도 선택하지 않은 것으로 가정합니다. `GetPossibleCount ([Initials])`	6이며, 선택 내용이 없고 **Initials**에 고유한 값이 6개이기 때문입니다.

데이터 사용 예:

```
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 - 차트 함수

GetSelectedCount()는 필드에서 선택한(녹색) 값의 수를 찾습니다.

구문:

```
GetSelectedCount (field_name [, include_excluded [, state_name]])
```

반환 데이터 유형: 정수

인수:

인수

인수	설명
field_name	측정할 데이터 범위가 포함된 필드입니다.
include_excluded	**True()**로 설정하면 카운트에 다른 필드의 선택 내용에 따라 현재 제외된 선택 값이 포함됩니다. False이거나 생략된 경우 해당 값이 포함되지 않습니다.
state_name	특정 시각화에 대해 선택한 대체 상태의 이름입니다. **state_name** 인수를 사용하는 경우 지정된 상태 이름과 연관된 선택 내용만 고려됩니다.

예 및 결과:

다음 예에서는 서로 다른 필터 창에 로드된 세 필드를 사용하며, 하나는 **First name** 이름, 하나는 **Initials**, 하나는 **Has cellphone**입니다.

예 및 결과

예	결과
First name에서 **John**을 선택한 것으로 가정합니다. `GetSelectedCount ([First name])`	1이며, **First name**에서 선택된 값이 하나이기 때문입니다.
First name에서 **John**을 선택한 것으로 가정합니다. `GetSelectedCount ([Initials])`	0이며, **Initials**에서 아무 값도 선택하지 않았기 때문입니다.
First name에서 아무 것도 선택하지 말고, **Initials**에서 모든 값을 선택한 다음 **Has cellphone**에서 **Yes**를 선택합니다. `GetSelectedCount ([Initials], True ())`	6입니다. **Initials** MC 및 PD의 선택 내용에서 **Has cellphone**이 **No**로 설정되었어도 결과는 그대로 6이며, 이는 인수 include_excluded가 True()로 설정되었기 때문입니다.

데이터 사용 예:

```
Names:
LOAD * inline [
First name|Last name|Initials|Has cellphone
John|Anderson|JA|Yes
```

```
Sue|Brown|SB|Yes
Mark|Carr|MC|No
Peter|Devonshire|PD|No
Jane|Elliot|JE|Yes
Peter|Franc|PF|Yes ] (delimiter is '|');
```

8.10 파일 함수

파일 함수(스크립트 표현식에서만 사용 가능)는 현재 읽고 있는 테이블 파일에 대한 정보를 반환합니다. 이 함수는 테이블 파일을 제외한 모든 데이터 소스에 대해 NULL을 반환합니다(예외: **ConnectString( )**).

파일 함수 개요

각 함수는 개요가 끝난 후에 더 자세히 설명합니다. 구문에서 함수 이름을 클릭하여 해당 함수에 대한 상세 설명에 즉시 액세스할 수도 있습니다.

Attribute

이 스크립트 함수는 다양한 미디어 파일의 메타 태그 값을 텍스트로 반환합니다. 지원되는 파일 형식은 MP3, WMA, WMV, PNG 및 JPG입니다. **filename** 파일이 존재하지 않거나, 지원되지 않는 파일 형식이거나, 이름이 **attributename**인 메타 태그가 포함되지 않은 경우에는 NULL이 반환됩니다.

```
Attribute (filename, attributename)
```

ConnectString

ConnectString() 함수는 ODBC에 대한 활성 데이터 연결 또는 OLE DB 연결의 이름을 반환합니다. **connect** 문이 실행되지 않았거나 **disconnect** 문 이후인 경우, 이 함수는 빈 문자열을 반환합니다.

```
ConnectString ()
```

FileBaseName

FileBaseName 함수는 경로 또는 확장명을 제외하고 현재 읽고 있는 테이블 파일의 이름이 포함된 문자열을 반환합니다.

```
FileBaseName ()
```

FileDir

FileDir 함수는 현재 읽고 있는 테이블 파일의 디렉터리에 대한 경로가 포함된 문자열을 반환합니다.

```
FileDir ()
```

FileExtension

FileExtension 함수는 현재 읽고 있는 테이블 파일의 확장명이 포함된 문자열을 반환합니다.

```
FileExtension ()
```

FileName

FileName 함수는 경로를 제외하되 확장명을 포함하여 현재 읽고 있는 테이블 파일의 이름이 포함된 문자열을 반환합니다.

```
FileName ()
```

FilePath

FilePath 함수는 현재 읽고 있는 테이블 파일에 대한 전체 경로가 포함된 문자열을 반환합니다.

```
FilePath ()
```

FileSize

FileSize 함수는 filename 파일의 바이트 단위 크기 또는 filename이 지정되지 않은 경우 현재 읽고 있는 테이블 파일의 바이트 단위 크기가 포함된 정수를 반환합니다.

```
FileSize ()
```

FileTime

FileTime 함수는 지정된 파일의 마지막 수정에 대한 UTC 서식의 타임스탬프를 반환합니다. 파일이 지정되지 않은 경우 함수는 현재 읽은 테이블 파일의 마지막 수정에 대한 타임스탬프(UTC)를 반환합니다.

```
FileTime ([ filename ])
```

GetFolderPath

GetFolderPath 함수는 Microsoft Windows *SHGetFolderPath* 함수의 값을 반환합니다. 이 함수는 Microsoft Windows 폴더의 이름을 입력으로 사용하여 이 폴더의 전체 경로를 반환합니다.

```
GetFolderPath ()
```

QvdCreateTime

이 스크립트 함수는 QVD 파일(있는 경우)의 XML 헤더 타임스탬프를 반환하며, 그렇지 않으면 NULL을 반환합니다. 타임스탬프에서 시간은 UTC로 제공됩니다.

```
QvdCreateTime (filename)
```

QvdFieldName

이 스크립트 함수는 QVD 파일 내의 필드 번호 **fieldno**의 이름을 반환합니다. 필드가 존재하지 않으면 NULL이 반환됩니다.

```
QvdFieldName (filename , fieldno)
```

QvdNoOfFields

이 스크립트 함수는 QVD 파일 내의 필드 수를 반환합니다.

```
QvdNoOfFields (filename)
```

QvdNoOfRecords

이 스크립트 함수는 QVD 파일 내의 현재 레코드 수를 반환합니다.

```
QvdNoOfRecords (filename)
```

QvdTableName

이 스크립트 함수는 QVD 파일에 저장된 테이블의 이름을 반환합니다.

```
QvdTableName (filename)
```

Attribute

이 스크립트 함수는 다양한 미디어 파일의 메타 태그 값을 텍스트로 반환합니다. 지원되는 파일 형식은 MP3, WMA, WMV, PNG 및 JPG입니다. **filename** 파일이 존재하지 않거나, 지원되지 않는 파일 형식이거나, 이름이 **attributename**인 메타 태그가 포함되지 않은 경우에는 NULL이 반환됩니다.

구문:

```
Attribute(filename, attributename)
```

많은 메타 태그를 읽을 수 있습니다. 이 항목의 예에는 지원되는 각 파일 형식에서 읽을 수 있는 태그가 나와 있습니다.

*관련 사양에 따라 파일에 저장된 메타 태그만 읽을 수 있으며(예: MP3 파일의 경우 ID2v3, JPG 파일의 경우 EXIF) **Windows 파일 탐색기**에 저장된 메타 정보는 읽을 수 없습니다.*

인수:

인수

인수	설명
filename	미디어 파일의 이름이며 필요할 경우 경로가 폴더 데이터 연결로 포함됩니다. ***'lib://Table Files/'*** 레거시 스크립팅 모드에서는 다음 경로 형식도 지원됩니다. • 절대 경로 ***c:\data*** • Qlik Sense 앱 작업 디렉터리에 대한 상대 경로. ***data***
attributename	메타 태그의 이름입니다.

예에서는 미디어 파일의 경로를 찾기 위해 **GetFolderPath** 함수를 사용합니다. **GetFolderPath**는 레거시 모드에서만 지원되므로 표준 모드 또는 Qlik Sense SaaS에서 이 함수를 사용할 때 **GetFolderPath**에 대한 참조를 lib:// 데이터 연결 경로로 바꿔야 합니다.

파일 시스템 액세스 제한 (page 1452)

Example 1: MP3 파일

이 스크립트는 *MyMusic* 폴더에서 사용 가능한 모든 MP3 메타 태그를 읽습니다.

```
// 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

이 스크립트는 *MyPictures* 폴더의 JPG 파일에서 사용 가능한 모든 EXIF 메타 태그를 읽습니다.

```
// Script to read Jpeg Exif meta tags
for each vExt in 'jpg', 'jpeg', 'jpe', 'jfif', 'jif', 'jfi'
for each vFoundFile in filelist( GetFolderPath('MyPictures') & '\*.'& vExt )
```

```
FileList:
LOAD FileLongName,
    subfield(FileLongName,'\',-1) as FileShortName,
    num(FileSize(FileLongName),'# ### ### ###',',',' ') as FileSize,
    FileTime(FileLongName) as FileTime,
    // ************   Exif Main (IFD0) Attributes   ************
    Attribute(FileLongName, 'ImageWidth') as ImageWidth,
    Attribute(FileLongName, 'ImageLength') as ImageLength,
    Attribute(FileLongName, 'BitsPerSample') as BitsPerSample,
    Attribute(FileLongName, 'Compression') as Compression,

    //  examples: 1=uncompressed, 2=CCITT, 3=CCITT 3, 4=CCITT 4,

    //5=LZW, 6=JPEG (old style), 7=JPEG, 8=Deflate, 32773=PackBits RLE,
    Attribute(FileLongName, 'PhotometricInterpretation') as PhotometricInterpretation,

    //  examples: 0=WhiteIsZero, 1=BlackIsZero, 2=RGB, 3=Palette, 5=CMYK, 6=YCbCr,
    Attribute(FileLongName, 'ImageDescription') as ImageDescription,
    Attribute(FileLongName, 'Make') as Make,
    Attribute(FileLongName, 'Model') as Model,
    Attribute(FileLongName, 'StripOffsets') as StripOffsets,
    Attribute(FileLongName, 'Orientation') as Orientation,

    //  examples: 1=TopLeft, 2=TopRight, 3=BottomRight, 4=BottomLeft,

    // 5=LeftTop, 6=RightTop, 7=RightBottom, 8=LeftBottom,
    Attribute(FileLongName, 'SamplesPerPixel') as SamplesPerPixel,
    Attribute(FileLongName, 'RowsPerStrip') as RowsPerStrip,
    Attribute(FileLongName, 'StripByteCounts') as StripByteCounts,
    Attribute(FileLongName, 'XResolution') as XResolution,
    Attribute(FileLongName, 'YResolution') as YResolution,
    Attribute(FileLongName, 'PlanarConfiguration') as PlanarConfiguration,

    //  examples: 1=chunky format, 2=planar format,
    Attribute(FileLongName, 'ResolutionUnit') as ResolutionUnit,

    //  examples: 1=none, 2=inches, 3=centimeters,
    Attribute(FileLongName, 'TransferFunction') as TransferFunction,
    Attribute(FileLongName, 'Software') as Software,
    Attribute(FileLongName, 'DateTime') as DateTime,
    Attribute(FileLongName, 'Artist') as Artist,
    Attribute(FileLongName, 'HostComputer') as HostComputer,
    Attribute(FileLongName, 'WhitePoint') as WhitePoint,
    Attribute(FileLongName, 'PrimaryChromaticities') as PrimaryChromaticities,
    Attribute(FileLongName, 'YCbCrCoefficients') as YCbCrCoefficients,
    Attribute(FileLongName, 'YCbCrSubSampling') as YCbCrSubSampling,
    Attribute(FileLongName, 'YCbCrPositioning') as YCbCrPositioning,

    //  examples: 1=centered, 2=co-sited,
    Attribute(FileLongName, 'ReferenceBlackWhite') as ReferenceBlackWhite,
    Attribute(FileLongName, 'Rating') as Rating,
    Attribute(FileLongName, 'RatingPercent') as RatingPercent,
    Attribute(FileLongName, 'ThumbnailFormat') as ThumbnailFormat,
```

```
    //  examples: 0=Raw Rgb, 1=Jpeg,
    Attribute(FileLongName, 'Copyright') as Copyright,
    Attribute(FileLongName, 'ExposureTime') as ExposureTime,
    Attribute(FileLongName, 'FNumber') as FNumber,
    Attribute(FileLongName, 'ExposureProgram') as ExposureProgram,

    //  examples: 0=Not defined, 1=Manual, 2=Normal program, 3=Aperture priority, 4=Shutter
priority,

    // 5=Creative program, 6=Action program, 7=Portrait mode, 8=Landscape mode, 9=Bulb,
    Attribute(FileLongName, 'ISOSpeedRatings') as ISOSpeedRatings,
    Attribute(FileLongName, 'TimeZoneOffset') as TimeZoneOffset,
    Attribute(FileLongName, 'SensitivityType') as SensitivityType,

    //  examples: 0=Unknown, 1=Standard output sensitivity (SOS), 2=Recommended exposure index
(REI),

    // 3=ISO speed, 4=Standard output sensitivity (SOS) and Recommended exposure index (REI),

    //5=Standard output sensitivity (SOS) and ISO Speed, 6=Recommended exposure index (REI)
and ISO Speed,

    // 7=Standard output sensitivity (SOS) and Recommended exposure index (REI) and ISO speed,
    Attribute(FileLongName, 'ExifVersion') as ExifVersion,
    Attribute(FileLongName, 'DateTimeOriginal') as DateTimeOriginal,
    Attribute(FileLongName, 'DateTimeDigitized') as DateTimeDigitized,
    Attribute(FileLongName, 'ComponentsConfiguration') as ComponentsConfiguration,

    //  examples: 1=Y, 2=Cb, 3=Cr, 4=R, 5=G, 6=B,
    Attribute(FileLongName, 'CompressedBitsPerPixel') as CompressedBitsPerPixel,
    Attribute(FileLongName, 'ShutterSpeedValue') as ShutterSpeedValue,
    Attribute(FileLongName, 'ApertureValue') as ApertureValue,
    Attribute(FileLongName, 'BrightnessValue') as BrightnessValue, //  examples: -1=Unknown,
    Attribute(FileLongName, 'ExposureBiasValue') as ExposureBiasValue,
    Attribute(FileLongName, 'MaxApertureValue') as MaxApertureValue,
    Attribute(FileLongName, 'SubjectDistance') as SubjectDistance,

    //  examples: 0=Unknown, -1=Infinity,
    Attribute(FileLongName, 'MeteringMode') as MeteringMode,

    //  examples: 0=Unknown, 1=Average, 2=CenterWeightedAverage, 3=Spot,

    // 4=MultiSpot, 5=Pattern, 6=Partial, 255=Other,
    Attribute(FileLongName, 'LightSource') as LightSource,

    //  examples: 0=Unknown, 1=Daylight, 2=Fluorescent, 3=Tungsten, 4=Flash, 9=Fine weather,

    // 10=Cloudy weather, 11=Shade, 12=Daylight fluorescent,

    // 13=Day white fluorescent, 14=Cool white fluorescent,

    // 15=White fluorescent, 17=Standard light A, 18=Standard light B, 19=Standard light C,

    // 20=D55, 21=D65, 22=D75, 23=D50, 24=ISO studio tungsten, 255=other light source,
    Attribute(FileLongName, 'Flash') as Flash,
```

```
    Attribute(FileLongName, 'FocalLength') as FocalLength,
    Attribute(FileLongName, 'SubjectArea') as SubjectArea,
    Attribute(FileLongName, 'MakerNote') as MakerNote,
    Attribute(FileLongName, 'UserComment') as UserComment,
    Attribute(FileLongName, 'SubSecTime') as SubSecTime,

    Attribute(FileLongName, 'SubsecTimeOriginal') as SubsecTimeOriginal,
    Attribute(FileLongName, 'SubsecTimeDigitized') as SubsecTimeDigitized,
    Attribute(FileLongName, 'XPTitle') as XPTitle,
    Attribute(FileLongName, 'XPComment') as XPComment,

    Attribute(FileLongName, 'XPAuthor') as XPAuthor,
    Attribute(FileLongName, 'XPKeywords') as XPKeywords,
    Attribute(FileLongName, 'XPSubject') as XPSubject,
    Attribute(FileLongName, 'FlashpixVersion') as FlashpixVersion,
    Attribute(FileLongName, 'ColorSpace') as ColorSpace, //  examples: 1=sRGB,
65535=Uncalibrated,
    Attribute(FileLongName, 'PixelXDimension') as PixelXDimension,
    Attribute(FileLongName, 'PixelYDimension') as PixelYDimension,
    Attribute(FileLongName, 'RelatedSoundFile') as RelatedSoundFile,

    Attribute(FileLongName, 'FocalPlaneXResolution') as FocalPlaneXResolution,
    Attribute(FileLongName, 'FocalPlaneYResolution') as FocalPlaneYResolution,
    Attribute(FileLongName, 'FocalPlaneResolutionUnit') as FocalPlaneResolutionUnit,

    //  examples: 1=None, 2=Inch, 3=Centimeter,
    Attribute(FileLongName, 'ExposureIndex') as ExposureIndex,
    Attribute(FileLongName, 'SensingMethod') as SensingMethod,

    //  examples: 1=Not defined, 2=One-chip color area sensor, 3=Two-chip color area sensor,

    // 4=Three-chip color area sensor, 5=Color sequential area sensor,

    // 7=Trilinear sensor, 8=Color sequential linear sensor,
    Attribute(FileLongName, 'FileSource') as FileSource,

    //  examples: 0=Other, 1=Scanner of transparent type,

    // 2=Scanner of reflex type, 3=Digital still camera,
    Attribute(FileLongName, 'SceneType') as SceneType,

    //  examples: 1=A directly photographed image,
    Attribute(FileLongName, 'CFAPattern') as CFAPattern,
    Attribute(FileLongName, 'CustomRendered') as CustomRendered,

    //  examples: 0=Normal process, 1=Custom process,
    Attribute(FileLongName, 'ExposureMode') as ExposureMode,

    //  examples: 0=Auto exposure, 1=Manual exposure, 2=Auto bracket,
    Attribute(FileLongName, 'WhiteBalance') as WhiteBalance,

    //  examples: 0=Auto white balance, 1=Manual white balance,
    Attribute(FileLongName, 'DigitalZoomRatio') as DigitalZoomRatio,
    Attribute(FileLongName, 'FocalLengthIn35mmFilm') as FocalLengthIn35mmFilm,
    Attribute(FileLongName, 'SceneCaptureType') as SceneCaptureType,
```

```
    //  examples: 0=Standard, 1=Landscape, 2=Portrait, 3=Night scene,
    Attribute(FileLongName, 'GainControl') as GainControl,

    //  examples: 0=None, 1=Low gain up, 2=High gain up, 3=Low gain down, 4=High gain down,
    Attribute(FileLongName, 'Contrast') as Contrast,

    //  examples: 0=Normal, 1=Soft, 2=Hard,
    Attribute(FileLongName, 'Saturation') as Saturation,

    //  examples: 0=Normal, 1=Low saturation, 2=High saturation,
    Attribute(FileLongName, 'Sharpness') as Sharpness,

    //  examples: 0=Normal, 1=Soft, 2=Hard,
    Attribute(FileLongName, 'SubjectDistanceRange') as SubjectDistanceRange,

    //  examples: 0=Unknown, 1=Macro, 2=Close view, 3=Distant view,
    Attribute(FileLongName, 'ImageUniqueID') as ImageUniqueID,
    Attribute(FileLongName, 'BodySerialNumber') as BodySerialNumber,
    Attribute(FileLongName, 'CMNT_GAMMA') as CMNT_GAMMA,
    Attribute(FileLongName, 'PrintImageMatching') as PrintImageMatching,
    Attribute(FileLongName, 'OffsetSchema') as OffsetSchema,

    // ************   Interoperability Attributes   ************
    Attribute(FileLongName, 'InteroperabilityIndex') as InteroperabilityIndex,
    Attribute(FileLongName, 'InteroperabilityVersion') as InteroperabilityVersion,
    Attribute(FileLongName, 'InteroperabilityRelatedImageFileFormat') as
InteroperabilityRelatedImageFileFormat,
    Attribute(FileLongName, 'InteroperabilityRelatedImageWidth') as
InteroperabilityRelatedImageWidth,
    Attribute(FileLongName, 'InteroperabilityRelatedImageLength') as
InteroperabilityRelatedImageLength,
    Attribute(FileLongName, 'InteroperabilityColorSpace') as InteroperabilityColorSpace,

    //  examples: 1=sRGB, 65535=Uncalibrated,
    Attribute(FileLongName, 'InteroperabilityPrintImageMatching') as
InteroperabilityPrintImageMatching,
    // ************   GPS Attributes   ************
    Attribute(FileLongName, 'GPSVersionID') as GPSVersionID,
    Attribute(FileLongName, 'GPSLatitudeRef') as GPSLatitudeRef,
    Attribute(FileLongName, 'GPSLatitude') as GPSLatitude,
    Attribute(FileLongName, 'GPSLongitudeRef') as GPSLongitudeRef,
    Attribute(FileLongName, 'GPSLongitude') as GPSLongitude,
    Attribute(FileLongName, 'GPSAltitudeRef') as GPSAltitudeRef,

    //  examples: 0=Above sea level, 1=Below sea level,
    Attribute(FileLongName, 'GPSAltitude') as GPSAltitude,
    Attribute(FileLongName, 'GPSTimeStamp') as GPSTimeStamp,
    Attribute(FileLongName, 'GPSSatellites') as GPSSatellites,
    Attribute(FileLongName, 'GPSStatus') as GPSStatus,
    Attribute(FileLongName, 'GPSMeasureMode') as GPSMeasureMode,
    Attribute(FileLongName, 'GPSDOP') as GPSDOP,
    Attribute(FileLongName, 'GPSSpeedRef') as GPSSpeedRef,

    Attribute(FileLongName, 'GPSSpeed') as GPSSpeed,
    Attribute(FileLongName, 'GPSTrackRef') as GPSTrackRef,
```

```
    Attribute(FileLongName, 'GPSTrack') as GPSTrack,
    Attribute(FileLongName, 'GPSImgDirectionRef') as GPSImgDirectionRef,
    Attribute(FileLongName, 'GPSImgDirection') as GPSImgDirection,
    Attribute(FileLongName, 'GPSMapDatum') as GPSMapDatum,
    Attribute(FileLongName, 'GPSDestLatitudeRef') as GPSDestLatitudeRef,

    Attribute(FileLongName, 'GPSDestLatitude') as GPSDestLatitude,
    Attribute(FileLongName, 'GPSDestLongitudeRef') as GPSDestLongitudeRef,
    Attribute(FileLongName, 'GPSDestLongitude') as GPSDestLongitude,
    Attribute(FileLongName, 'GPSDestBearingRef') as GPSDestBearingRef,
    Attribute(FileLongName, 'GPSDestBearing') as GPSDestBearing,
    Attribute(FileLongName, 'GPSDestDistanceRef') as GPSDestDistanceRef,

    Attribute(FileLongName, 'GPSDestDistance') as GPSDestDistance,
    Attribute(FileLongName, 'GPSProcessingMethod') as GPSProcessingMethod,
    Attribute(FileLongName, 'GPSAreaInformation') as GPSAreaInformation,
    Attribute(FileLongName, 'GPSDateStamp') as GPSDateStamp,
    Attribute(FileLongName, 'GPSDifferential') as GPSDifferential;

    //  examples: 0=No correction, 1=Differential correction,
LOAD @1:n as FileLongName Inline "$(vFoundFile)" (fix, no labels);
Next vFoundFile
Next vExt
```

Example 3: Windows 미디어 파일

이 스크립트는 *MyMusic* 폴더에서 사용 가능한 모든 WMA/WMV ASF 메타 태그를 읽습니다.

```
/ 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

이 스크립트는 *MyPictures* 폴더에서 사용 가능한 모든 PNG 메타 태그를 읽습니다.

```
// 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

ConnectString() 함수는 ODBC에 대한 활성 데이터 연결 또는 OLE DB 연결의 이름을 반환합니다. **connect** 문이 실행되지 않았거나 **disconnect** 문 이후인 경우, 이 함수는 빈 문자열을 반환합니다.

구문:

```
ConnectString()
```

예 및 결과:

스크립팅 예

예	결과

`LIB CONNECT TO 'Tutorial ODBC';`
 	
`ConnectString:`
 	
`Load ConnectString() as ConnectString AutoGenerate 1;`
	ConnectString 필드에서 'Tutorial ODBC'를 반환합니다.
	
	이 예에서는 Tutorial ODBC라는 이름의 사용 가능한 데이터 연결이 있는 것으로 가정합니다.

FileBaseName

FileBaseName 함수는 경로 또는 확장명을 제외하고 현재 읽고 있는 테이블 파일의 이름이 포함된 문자열을 반환합니다.

구문:

```
FileBaseName()
```

예 및 결과:

스크립팅 예

예	결과
LOAD *, filebasename( ) as X from C:\UserFiles\abc.txt	읽은 각 레코드의 X 필드에서 'abc'를 반환합니다.

FileDir

FileDir 함수는 현재 읽고 있는 테이블 파일의 디렉터리에 대한 경로가 포함된 문자열을 반환합니다.

구문:

```
FileDir()
```

이 기능은 표준 모드의 폴더 데이터 연결만 지원합니다.

예 및 결과:

스크립팅 예

예	결과
Load *, filedir( ) as X from C:\UserFiles\abc.txt	읽은 각 레코드의 X 필드에서 'C:\UserFiles'를 반환합니다.

FileExtension

FileExtension 함수는 현재 읽고 있는 테이블 파일의 확장명이 포함된 문자열을 반환합니다.

구문:

```
FileExtension()
```

예 및 결과:

스크립팅 예

예	결과
LOAD *, FileExtension( ) as X from C:\UserFiles\abc.txt	읽은 각 레코드의 X 필드에서 'txt'를 반환합니다.

FileName

FileName 함수는 경로를 제외하되 확장명을 포함하여 현재 읽고 있는 테이블 파일의 이름이 포함된 문자열을 반환합니다.

구문:

```
FileName()
```

예 및 결과:

스크립팅 예

예	결과
LOAD *, FileName( ) as X from C:\UserFiles\abc.txt	읽은 각 레코드의 X 필드에서 'abc.txt'를 반환합니다.

FilePath

FilePath 함수는 현재 읽고 있는 테이블 파일에 대한 전체 경로가 포함된 문자열을 반환합니다.

구문:

```
FilePath()
```

이 기능은 표준 모드의 폴더 데이터 연결만 지원합니다.

예 및 결과:

스크립팅 예

예	결과
Load *, FilePath( ) as X from C:\UserFiles\abc.txt	읽은 각 레코드의 X 필드에서 'C:\UserFiles\abc.txt'를 반환합니다.

FileSize

FileSize 함수는 filename 파일의 바이트 단위 크기 또는 filename이 지정되지 않은 경우 현재 읽고 있는 테이블 파일의 바이트 단위 크기가 포함된 정수를 반환합니다.

구문:
FileSize([filename]**)**

인수:

인수

인수	설명
filename	폴더 또는 웹 파일 데이터 연결로 필요한 경우 경로를 포함하여 파일 이름 파일 이름을 지정하지 않으면 현재 읽고 있는 테이블 파일이 사용됩니다. ***'lib://Table Files/'*** 레거시 스크립팅 모드에서는 다음 경로 형식도 지원됩니다. • 절대 경로 ***c:\data*** • Qlik Sense 앱 작업 디렉터리에 대한 상대 경로. ***data*** • 인터넷 또는 인트라넷상의 위치를 가리키는 URL 주소(HTTP 또는 FTP) ***http://www.qlik.com***

예 및 결과:

스크립팅 예

예	결과
LOAD *, FileSize( ) as X from abc.txt;	읽은 각 레코드의 X 필드에서 지정된 파일(abc.txt)의 크기를 정수로 반환합니다.
FileSize('lib://DataFiles/xyz.xls')	xyz.xls 파일의 크기를 반환합니다.

FileTime

FileTime 함수는 지정된 파일의 마지막 수정에 대한 UTC 서식의 타임스탬프를 반환합니다. 파일이 지정되지 않은 경우 함수는 현재 읽은 테이블 파일의 마지막 수정에 대한 타임스탬프(UTC)를 반환합니다.

구문:
FileTime([filename]**)**

인수:

인수

인수	설명
filename	폴더 또는 웹 파일 데이터 연결에 해당하는 파일의 이름이며, 필요할 경우 경로가 포함됩니다. ***'lib://Table Files/'*** 레거시 스크립팅 모드에서는 다음 경로 형식도 지원됩니다. • 절대 경로 ***c:\data*** • Qlik Sense 앱 작업 디렉터리에 대한 상대 경로. ***data*** • 인터넷 또는 인트라넷상의 위치를 가리키는 URL 주소(HTTP 또는 FTP) ***http://www.qlik.com***

예 및 결과:

스크립팅 예

예	결과
`LOAD *, FileTime( ) as X from abc.txt;`	읽은 각 레코드의 X 필드에 있는 파일(abc.txt)의 마지막 수정 시간 스탬프를 반환합니다.
`FileTime( 'xyz.xls' )`	Xyz.xls 파일의 최종 수정 타임스탬프를 반환합니다.

GetFolderPath

GetFolderPath 함수는 Microsoft Windows *SHGetFolderPath* 함수의 값을 반환합니다. 이 함수는 Microsoft Windows 폴더의 이름을 입력으로 사용하여 이 폴더의 전체 경로를 반환합니다.

표준 모드에서는 이 기능이 지원되지 않습니다.

구문:

```
GetFolderPath(foldername)
```

인수:

인수

인수	설명
foldername	Microsoft Windows 폴더의 이름입니다. 폴더 이름에는 어떠한 공백도 포함되지 않아야 합니다. 따라서 Windows Explorer에 표시되는 폴더 이름에 포함된 모든 공백이 해당 폴더 이름에서 제거되어야 합니다. 예: *MyMusic* *MyDocuments*

예 및 결과:

이 예는 Microsoft Windows 폴더인 *MyMusic*, *MyPictures* 및 *Windows*의 경로를 얻는 것이 목적입니다. 이 예제 스크립트를 앱에 추가하고 다시 로드하십시오.

```
LOAD
 GetFolderPath('MyMusic') as MyMusic,
 GetFolderPath('MyPictures') as MyPictures,
 GetFolderPath('Windows') as Windows
AutoGenerate 1;
```

앱을 다시 로드하면 *MyMusic*, *MyPictures* 및 *Windows* 필드가 데이터 모델에 추가됩니다. 각 필드에는 입력에 정의된 폴더의 경로가 포함됩니다. 예:

- *C:\Users\smu\Music* for the folder *MyMusic*
- *C:\Users\smu\Pictures* for the folder *MyPictures*
- *C:\Windows* for the folder *Windows*

QvdCreateTime

이 스크립트 함수는 QVD 파일(있는 경우)의 XML 헤더 타임스탬프를 반환하며, 그렇지 않으면 NULL을 반환합니다. 타임스탬프에서 시간은 UTC로 제공됩니다.

구문:

```
QvdCreateTime(filename)
```

인수:

인수

인수	설명
filename	폴더 또는 웹 데이터 연결에 해당하는 QVD 파일의 이름이며, 필요할 경우 경로가 포함됩니다. ***'lib://Table Files/'*** 레거시 스크립팅 모드에서는 다음 경로 형식도 지원됩니다. • 절대 경로 ***c:\data*** • Qlik Sense 앱 작업 디렉터리에 대한 상대 경로. ***data*** • 인터넷 또는 인트라넷상의 위치를 가리키는 URL 주소(HTTP 또는 FTP) ***http://www.qlik.com***

```
QvdCreateTime('MyFile.qvd')

QvdCreateTime('C:\MyDir\MyFile.qvd')

QvdCreateTime('lib://DataFiles/MyFile.qvd')
```

QvdFieldName

이 스크립트 함수는 QVD 파일 내의 필드 번호 **fieldno**의 이름을 반환합니다. 필드가 존재하지 않으면 NULL이 반환됩니다.

구문:

```
QvdFieldName(filename , fieldno)
```

인수:

인수

인수	설명
filename	폴더 또는 웹 데이터 연결에 해당하는 QVD 파일의 이름이며, 필요할 경우 경로가 포함됩니다. ***'lib://Table Files/'*** 레거시 스크립팅 모드에서는 다음 경로 형식도 지원됩니다. • 절대 경로 ***c:\data*** • Qlik Sense 앱 작업 디렉터리에 대한 상대 경로. ***data*** • 인터넷 또는 인트라넷상의 위치를 가리키는 URL 주소(HTTP 또는 FTP) ***http://www.qlik.com***
fieldno	QVD 파일에 포함된 테이블 내 필드의 번호입니다.

```
QvdFieldName ('MyFile.qvd', 5)

QvdFieldName ('C:\MyDir\MyFile.qvd', 5)

QvdFieldName ('lib://DataFiles/MyFile.qvd', 5)
```
세 가지 예 모두 QVD 파일에 포함된 테이블의 다섯 번째 필드 이름을 반환합니다.

QvdNoOfFields

이 스크립트 함수는 QVD 파일 내의 필드 수를 반환합니다.

구문:

```
QvdNoOfFields(filename)
```

인수:

인수

인수	설명
filename	폴더 또는 웹 데이터 연결에 해당하는 QVD 파일의 이름이며, 필요할 경우 경로가 포함됩니다. ***'lib://Table Files/'*** 레거시 스크립팅 모드에서는 다음 경로 형식도 지원됩니다. • 절대 경로 ***c:\data*** • Qlik Sense 앱 작업 디렉터리에 대한 상대 경로. ***data*** • 인터넷 또는 인트라넷상의 위치를 가리키는 URL 주소(HTTP 또는 FTP) ***http://www.qlik.com***

```
QvdNoOfFields ('MyFile.qvd')

QvdNoOfFields ('C:\MyDir\MyFile.qvd')

QvdNoOfFields ('lib://DataFiles/MyFile.qvd')
```

QvdNoOfRecords

이 스크립트 함수는 QVD 파일 내의 현재 레코드 수를 반환합니다.

구문:

```
QvdNoOfRecords(filename)
```

인수:

인수

인수	설명
filename	폴더 또는 웹 데이터 연결에 해당하는 QVD 파일의 이름이며, 필요할 경우 경로가 포함됩니다. ***'lib://Table Files/'*** 레거시 스크립팅 모드에서는 다음 경로 형식도 지원됩니다. • 절대 경로 ***c:\data*** • Qlik Sense 앱 작업 디렉터리에 대한 상대 경로. ***data*** • 인터넷 또는 인트라넷상의 위치를 가리키는 URL 주소(HTTP 또는 FTP) ***http://www.qlik.com***

```
QvdNoOfRecords ('MyFile.qvd')

QvdNoOfRecords ('C:\MyDir\MyFile.qvd')

QvdNoOfRecords ('lib://DataFiles/MyFile.qvd')
```

QvdTableName

이 스크립트 함수는 QVD 파일에 저장된 테이블의 이름을 반환합니다.

구문:

```
QvdTableName(filename)
```

인수:

인수

인수	설명
filename	폴더 또는 웹 데이터 연결에 해당하는 QVD 파일의 이름이며, 필요할 경우 경로가 포함됩니다. ***'lib://Table Files/'*** 레거시 스크립팅 모드에서는 다음 경로 형식도 지원됩니다. • 절대 경로 ***c:\data*** • Qlik Sense 앱 작업 디렉터리에 대한 상대 경로. ***data*** • 인터넷 또는 인트라넷상의 위치를 가리키는 URL 주소(HTTP 또는 FTP) ***http://www.qlik.com***

```
QvdTableName ('MyFile.qvd')

QvdTableName ('C:\MyDir\MyFile.qvd')

QvdTableName ('lib://data\MyFile.qvd')
```

8.11 재무 함수

재무 함수는 데이터 로드 스크립트와 차트 표현식에서 납입금 및 이자율을 계산하는 데 사용할 수 있습니다.
모든 인수에서 납입된 현금은 음수로 표시됩니다. 수납된 현금은 양수로 표시됩니다.
아래에 재무 함수에 사용되는 인수가 나와 있습니다(**range**-로 시작되는 함수 제외).

*모든 재무 함수는 **rate** 및 **nper**의 단위를 지정할 때 일관성을 유지하는 것이 매우 중요합니다. 연이자 6%의 5년 대출에 대해 월 단위로 결제가 이루어지는 경우 **rate**에는 0.005(6%/12)를 사용하고 **nper**에는 60(5*12)을 사용하십시오. 동일한 대출에 대해 연 단위로 결제가 이루어지는 경우 **rate**에는 6%를 사용하고 **nper**에는 5를 사용하십시오.*

재무 함수 개요

각 함수는 개요가 끝난 후에 더 자세히 설명합니다. 구문에서 함수 이름을 클릭하여 해당 함수에 대한 상세 설명에 즉시 액세스할 수도 있습니다.

FV
이 함수는 주기적, 고정적 납입 및 단순 연 이자를 기준으로 한 미래 투자 가치를 반환합니다.

```
FV (rate, nper, pmt [ ,pv [ , type ] ])
```

nPer
이 함수는 주기적, 고정적 납입 및 고정 이자율을 기준으로 한 투자의 납입 회차를 반환합니다.

```
nPer (rate, pmt, pv [ ,fv [ , type ] ])
```

Pmt
이 함수는 주기적, 고정적 납입 및 고정 이자율을 기준으로 한 대출의 납입금을 반환합니다. 이 값은 전체 연금 기간 동안 변경되지 않습니다. 납입금은 음수(예: -20)로 지정됩니다.

```
Pmt (rate, nper, pv [ ,fv [ , type ] ] )
```

PV
이 함수는 투자의 현재 가치를 반환합니다.

```
PV (rate, nper, pmt [ ,fv [ , type ] ])
```

Rate
이 함수는 연금의 기간별 이자율을 반환합니다. 결과의 기본 숫자 서식은 **Fix** 소수점 아래 두 자리와 %입니다.

```
Rate (nper, pmt , pv [ ,fv [ , type ] ])
```

BlackAndSchole

Black and Scholes 모델은 금융 시장에서 파생 상품에 사용되는 수학적 모델입니다. 이 공식은 옵션의 이론적 가치를 계산합니다. Qlik Sense에서 **BlackAndSchole** 함수는 Black and Scholes 비수정 공식(유럽식 스타일 옵션)에 따라 값을 반환합니다.

```
BlackAndSchole(strike , time_left , underlying_price , vol , risk_free_rate , type)
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
strike	주식의 향후 구매 가격입니다.

인수	설명
time_left	남은 기간을 나타내는 숫자입니다.
underlying_price	주식의 현재 가치입니다.
vol	기간에 대해 10진수 형태의 백분율로 표현되는 변동성(주가)입니다.
risk_free_rate	기간에 대해 10진수 형태의 백분율로 표현되는 무위험 수익율입니다.
call_or_put	옵션의 유형: 콜옵션의 경우 'c', 'call' 또는 0이 아닌 숫자 값 풋옵션의 경우 'p', 'put' 또는 0

제한 사항:

strike, time_left 및 underlying_price 값은 0보다 커야 합니다.

vol 및 risk_free_rate 값은 0보다 작거나 0보다 커야 합니다.

예 및 결과:

스크립팅 예

예	결과
`BlackAndScholes(130, 4, 68.5, 0.4, 0.04, 'call')` 4년 동안 현재 가치가 68.5인 주식 130주를 구매하는 옵션의 이론적 가격이 계산됩니다. 이 수식에서는 연간 0.4(40%)의 변동성 및 0.04(4%)의 무위험 수익율을 사용합니다.	11.245를 반환합니다.

FV

이 함수는 주기적, 고정적 납입 및 단순 연 이자를 기준으로 한 미래 투자 가치를 반환합니다.

구문:

```
FV(rate, nper, pmt [ ,pv [ , type ] ])
```

반환 데이터 유형: 숫자 기본적으로 결과는 통화로 서식이 지정됩니다..

인수:

인수

인수	설명
rate	기간별 이자율입니다.
nper	연금의 총 납입 횟수입니다.
pmt	각 회차당 납입금입니다. 이 값은 전체 연금 기간 동안 변경되지 않습니다. 납입금은 음수(예: -20)로 지정됩니다.

인수	설명
pv	일련의 미래 납입금을 현재 기준으로 환산한 현재 가치 또는 총액입니다. **pv**를 생략한 경우 0으로 가정됩니다.
type	기간 말에 납입이 이루어지는 경우 0, 기간 초에 납입이 이루어지는 경우 1로 설정해야 합니다. **type**를 생략한 경우 0으로 가정됩니다.

예 및 결과:

스크립팅 예

예	결과
새로 산 가전 제품의 구매 대금을 36개월간 $20씩 월부로 납입하고 있습니다. 이자율은 연간 6%입니다. 청구서는 매월 말일에 옵니다. 마지막 청구서 대금을 납입했을 때 납입된 총액은 얼마입니까? `FV(0.005,36,-20)`	$786.72를 반환합니다.

nPer

이 함수는 주기적, 고정적 납입 및 고정 이자율을 기준으로 한 투자의 납입 회차를 반환합니다.

구문:

```
nPer(rate, pmt, pv [ ,fv [ , type ] ])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
rate	기간별 이자율입니다.
nper	연금의 총 납입 횟수입니다.
pmt	각 회차당 납입금입니다. 이 값은 전체 연금 기간 동안 변경되지 않습니다. 납입금은 음수(예: -20)로 지정됩니다.
pv	일련의 미래 납입금을 현재 기준으로 환산한 현재 가치 또는 총액입니다. **pv**를 생략한 경우 0으로 가정됩니다.
fv	마지막 납입이 완료된 후 달성하고자 하는 미래 가치 또는 현금 잔고입니다. **fv**를 생략한 경우 0으로 가정됩니다.
type	기간 말에 납입이 이루어지는 경우 0, 기간 초에 납입이 이루어지는 경우 1로 설정해야 합니다. **type**를 생략한 경우 0으로 가정됩니다.

예 및 결과:

스크립팅 예

예	결과
가전 제품을 $20씩 월부로 판매하려고 합니다. 이자율은 연간 6%입니다. 청구서는 매월 말일에 옵니다. 마지막 청구서의 납입이 완료된 후 수납된 금액의 가치가 $800가 되도록 하려면 납입 회차를 얼마로 해야 합니까? `nPer(0.005,-20,0,800)`	36.56을 반환합니다.

Pmt

이 함수는 주기적, 고정적 납입 및 고정 이자율을 기준으로 한 대출의 납입금을 반환합니다. 이 값은 전체 연금 기간 동안 변경되지 않습니다. 납입금은 음수(예: -20)로 지정됩니다.

```
Pmt(rate, nper, pv [ ,fv [ , type ] ] )
```

반환 데이터 유형: 숫자 기본적으로 결과는 통화로 서식이 지정됩니다..

대출 기간 동안 납입되는 총액을 계산하려면 반환된 **pmt** 값에 **nper**을 곱하십시오.

인수:

인수

인수	설명
rate	기간별 이자율입니다.
nper	연금의 총 납입 횟수입니다.
pv	일련의 미래 납입금을 현재 기준으로 환산한 현재 가치 또는 총액입니다. **pv**를 생략한 경우 0으로 가정됩니다.
fv	마지막 납입이 완료된 후 달성하고자 하는 미래 가치 또는 현금 잔고입니다. **fv**를 생략한 경우 0으로 가정됩니다.
type	기간 말에 납입이 이루어지는 경우 0, 기간 초에 납입이 이루어지는 경우 1로 설정해야 합니다. **type**를 생략한 경우 0으로 가정됩니다.

예 및 결과:

스크립팅 예

예	결과
다음 수식은 8개월 안에 상환해야 하는 연간 이자율 10%의 $20,000 대출에 해당하는 매월 납입금을 반환합니다. `Pmt(0.1/12,8,20000)`	-$2,594.66를 반환합니다.
동일한 대출에서 기간 초에 납입이 이루어져야 한다면 납입금은 다음과 같습니다. `Pmt(0.1/12,8,20000,0,1)`	-$2,573.21를 반환합니다.

PV

이 함수는 투자의 현재 가치를 반환합니다.

```
PV(rate, nper, pmt [ ,fv [ , type ] ])
```

반환 데이터 유형: 숫자 기본적으로 결과는 통화로 서식이 지정됩니다..

현재 가치란 일련의 미래 납입금을 현재 기준으로 환산한 총액입니다. 예를 들어 금전을 대출할 때 대출액은 채권자에 대한 현재 가치입니다.

인수:

인수

인수	설명
rate	기간별 이자율입니다.
nper	연금의 총 납입 횟수입니다.
pmt	각 회차당 납입금입니다. 이 값은 전체 연금 기간 동안 변경되지 않습니다. 납입금은 음수(예: -20)로 지정됩니다.
fv	마지막 납입이 완료된 후 달성하고자 하는 미래 가치 또는 현금 잔고입니다. **fv**를 생략한 경우 0으로 가정됩니다.
type	기간 말에 납입이 이루어지는 경우 0, 기간 초에 납입이 이루어지는 경우 1로 설정해야 합니다. **type**를 생략한 경우 0으로 가정됩니다.

예 및 결과:

스크립팅 예

예	결과
이자율을 7%라고 할 때 5년 동안 매월 말일에 $100가 수납된다면 부채의 현재 가치는 얼마입니까? `PV(0.07/12,12*5,-100,0,0)`	$5,050.20를 반환합니다.

Rate

이 함수는 연금의 기간별 이자율을 반환합니다. 결과의 기본 숫자 서식은 **Fix** 소수점 아래 두 자리와 %입니다.

구문:

```
Rate(nper, pmt , pv [ ,fv [ , type ] ])
```

반환 데이터 유형: 숫자

rate는 반복적으로 계산되며 0 또는 추가적인 해가 있을 수 있습니다. **rate**의 연속적인 결과가 수렴되지 않을 경우 NULL 값이 반환됩니다.

인수:

인수

인수	설명
nper	연금의 총 납입 횟수입니다.
pmt	각 회차당 납입금입니다. 이 값은 전체 연금 기간 동안 변경되지 않습니다. 납입금은 음수(예: -20)로 지정됩니다.
pv	일련의 미래 납입금을 현재 기준으로 환산한 현재 가치 또는 총액입니다. **pv**를 생략한 경우 0으로 가정됩니다.
fv	마지막 납입이 완료된 후 달성하고자 하는 미래 가치 또는 현금 잔고입니다. **fv**를 생략한 경우 0으로 가정됩니다.
type	기간 말에 납입이 이루어지는 경우 0, 기간 초에 납입이 이루어지는 경우 1로 설정해야 합니다. **type**를 생략한 경우 0으로 가정됩니다.

예 및 결과:

스크립팅 예

예	결과
월 납입금이 $300인 5년간의 $10,000 연금 대출의 이자율은 얼마입니까? `Rate(60,-300,10000)`	2.00%를 반환합니다.

8.12 서식 지정 함수

서식 지정 함수는 입력 숫자 필드 또는 표현식에 표시 서식을 적용합니다. 데이터 서식에 따라 소수점 구분 기호, 천 단위 구분 기호 등에 사용할 문자를 지정할 수 있습니다.

모든 함수는 문자열과 숫자 값이 모두 포함된 이중 값을 반환하지만 숫자-문자열 변환을 수행하는 것과 유사하게 생각할 수 있습니다. **Dual()**은 특별한 사례이지만 다른 서식 지정 함수는 입력 표현식의 숫자 값을 사용하여 숫자를 나타내는 문자열을 생성합니다.

반면, 해석 함수는 반대로 작업을 수행합니다. 결과 숫자의 서식을 지정하는 문자열 표현식을 숫자로 평가합니다.

이 함수는 데이터 로드 스크립트와 차트 표현식에서 모두 사용할 수 있습니다.

모든 숫자 표현에는 소수점 구분 기호로 소수점이 사용됩니다.

서식 지정 함수 개요

각 함수는 개요가 끝난 후에 더 자세히 설명합니다. 구문에서 함수 이름을 클릭하여 해당 함수에 대한 상세 설명에 즉시 액세스할 수도 있습니다.

ApplyCodepage

ApplyCodepage()는 표현식에 지정된 필드 또는 텍스트에 다른 코드 페이지 문자 집합을 적용합니다. **codepage** 인수는 숫자 형식이어야 합니다.

```
ApplyCodepage (text, codepage)
```

Date

Date()는 데이터 로드 스크립트, 운영 체제 또는 서식 문자열(지정된 경우)로 설정된 서식을 사용하여 표현식의 서식을 날짜로 지정합니다.

```
Date (number[, format])
```

Dual

Dual()은 숫자와 문자열을 단일 레코드로 결합합니다. 이러한 레코드의 숫자 표현은 정렬 및 계산 목적에 사용할 수 있으며, 문자열 값은 표시 목적으로 사용할 수 있습니다.

```
Dual (text, number)
```

Interval

Interval()은 데이터 로드 스크립트, 운영 체제 또는 서식 문자열(지정된 경우)에서 시스템 변수의 서식을 사용하여 숫자를 시간 간격으로 서식 지정합니다.

```
Interval (number[, format])
```

Money

Money()는 데이터 로드 스크립트 또는 운영 체제(서식 문자열을 지정하지 않은 경우)에 설정된 시스템 변수에 지정된 서식 및 선택적 소수점 구분 기호와 천 단위 구분 기호를 사용하여 표현식을 숫자 화폐 값으로 지정합니다.

```
Money (number[, format[, dec_sep [, thou_sep]]])
```

Num

Num()은 숫자의 형식을 지정합니다. 즉, 입력의 숫자 값을 변환하여 두 번째 매개 변수에 지정된 형식을 사용하여 텍스트를 표시합니다. 두 번째 매개 변수를 생략하면 데이터 로드 스크립트에 설정된 소수 및 천 단위 구분 기호를 사용합니다. 사용자 지정 소수점 구분 기호 및 천 단위 구분 기호는 옵션 파라메타입니다.

```
Num (number[, format[, dec_sep [, thou_sep]]])
```

Time

Time()은 데이터 로드 스크립트 또는 운영 체제(서식 문자열을 지정하지 않은 경우)의 시스템 변수에 설정된 시간 서식을 사용하여 표현식을 시간 값으로 서식 지정합니다.

```
Time (number[, format])
```

Timestamp

TimeStamp()는 데이터 로드 스크립트 또는 운영 체제(서식 문자열을 지정하지 않은 경우)의 시스템 변수에 설정된 타임스탬프 서식을 사용하여 표현식을 날짜 및 시간 값으로 서식 지정합니다.

```
Timestamp (number[, format])
```

관련 항목:

- *해석 함수 (page 1217)*

ApplyCodepage

ApplyCodepage()는 표현식에 지정된 필드 또는 텍스트에 다른 코드 페이지 문자 집합을 적용합니다. **codepage** 인수는 숫자 형식이어야 합니다.

> *ApplyCodepage는 차트 표현식에서 사용할 수도 있지만 데이터 로드 편집기에서 스크립트 함수로 사용되는 것이 더 일반적입니다. 예를 들어 사용자가 제어할 수 없는 다른 문자 집합으로 저장되었을 가능성이 있는 파일을 로드한 경우, 필요한 문자 집합을 나타내는 코드 페이지를 적용할 수 있습니다.*

구문:

```
ApplyCodepage(text, codepage)
```

반환 데이터 유형: 문자열

인수:

인수

인수	설명
text	**codepage** 인수를 통해 지정된 다른 코드 페이지를 적용할 필드 또는 텍스트입니다.
codepage	**text**를 통해 지정된 필드 또는 표현식에 적용할 코드 페이지를 나타내는 숫자입니다.

예 및 결과:

스크립팅 예

예	결과
`LOAD ApplyCodepage(ROWX,1253) as GreekProduct, ApplyCodepage (ROWY, 1255) as HebrewProduct, ApplyCodepage (ROWZ, 65001) as EnglishProduct; SQL SELECT ROWX, ROWY, ROWZ From Products;`	SQL에서 로드하는 경우 소스에는 UTF-8 형식의 다양한 문자 집합(키릴 문자, 히브리어 등)이 혼합되어 있을 수 있습니다. 이런 경우 각 행마다 다른 코드 페이지를 적용하여 한 행씩 로드해야 할 수 있습니다. **codepage** 값 1253은 Windows 그리스어 문자 집합을 나타내고 값 1255는 히브리어, 값 65001은 표준 라틴 UTF-8 문자를 나타냅니다.

관련 항목: *문자 집합 (page 162)*

Date

Date()는 데이터 로드 스크립트, 운영 체제 또는 서식 문자열(지정된 경우)로 설정된 서식을 사용하여 표현식의 서식을 날짜로 지정합니다.

구문:

```
Date(number[, format])
```

반환 데이터 유형: dual

인수:

인수

인수	설명
number	서식 지정할 숫자입니다.
format	결과 문자열의 서식을 설명하는 문자열입니다. 서식 문자열을 지정하지 않으면 데이터 로드 스크립트의 시스템 변수 또는 운영 체제에 설정된 날짜 서식이 사용됩니다.

예 및 결과:
아래 예에서는 다음과 같은 기본 설정이 사용된다고 가정합니다.

- 날짜 설정 1: YY-MM-DD
- 날짜 설정 2: M/D/YY

```
Date( A )
```
여기서, A=35648

결과 테이블

결과	설정 1	설정 2
문자열:	97-08-06	8/6/97
숫자:	35648	35648

```
Date( A, 'YY.MM.DD' )
```
여기서, A=35648

결과 테이블

결과	설정 1	설정 2
문자열:	97.08.06	97.08.06
숫자:	35648	35648

```
Date( A, 'DD.MM.YYYY' )
```
여기서, A=35648.375

결과 테이블

결과	설정 1	설정 2
문자열:	06.08.1997	06.08.1997
숫자:	35648.375	35648.375

```
Date( A, 'YY.MM.DD' )
```
여기서, A=8/6/97

결과 테이블

결과	설정 1	설정 2
문자열:	NULL(없음)	97.08.06
숫자:	NULL	35648

Dual

Dual()은 숫자와 문자열을 단일 레코드로 결합합니다. 이러한 레코드의 숫자 표현은 정렬 및 계산 목적에 사용할 수 있으며, 문자열 값은 표시 목적으로 사용할 수 있습니다.

구문:

```
Dual(text, number)
```

반환 데이터 유형: dual

인수:

인수

인수	설명
text	숫자 인수와 조합하여 사용할 문자열 값입니다.
number	문자열 인수의 문자열과 조합하여 사용할 숫자입니다.

Qlik Sense에서 모든 필드 값은 잠재적으로 이중 값입니다. 따라서 필드 값에 숫자 값과 텍스트 값을 모두 포함할 수 있습니다. 예제는 숫자 값 40908, 텍스트 표현 '2011-12-31'을 가질 수 있는 날짜입니다.

한 필드에 읽어들인 여러 데이터 항목이 문자열 표현은 서로 다르지만 동일하게 유효한 숫자 표현을 가진 경우 모든 데이터 항목이 첫 번째로 발견되는 문자열 표현을 공유합니다.

***dual** 함수는 일반적으로 다른 데이터를 관련 필드로 읽기 전에 필터 창 등에 표시될 첫 번째 문자열 표현을 만들기 위해 스크립트 초반에 사용됩니다.*

예 및 결과:

스크립팅 예

예	설명

다음 예를 스크립트에 추가하고 실행합니다.
`Load dual ( NameDay,NumDay ) as DayOfWeek inline`
`[ NameDay,NumDay`
`Monday,0`
`Tuesday,1`
`Wednesday,2`
`Thursday,3`
`Friday,4`
`Saturday,5`
`Sunday,6 ];`
	DayOfWeek 필드는 시각화에서 차원 등으로 사용할 수 있습니다. 요일이 있는 테이블은 사전순이 아닌, 정확한 번호 순서로 자동 정렬됩니다.

`Load Dual('Q' & Ceil(Month (Now())/3), Ceil(Month(Now ())/3)) as Quarter AutoGenerate 1;`	이 예에서는 현재 분기를 찾습니다. **Now()** 함수가 해당 연도의 처음 3개월 내에 실행되면 Q1로 표시되고 두 번째 3개월에 대해서는 Q2가 표시됩니다. 하지만 정렬에 사용하는 경우 Quarter 필드는 숫자 값 1 ~ 4로 동작합니다.
`Dual('Q' & Ceil(Month (Date)/3), Ceil(Month (Date)/3)) as Quarter`	이전 예에서처럼 Quarter 필드를 텍스트 값 'Q1' ~ 'Q4'로 만들고, 숫자 값 1 ~ 4를 할당합니다. 스크립트에서 사용하기 위해서는 Date의 값을 로드해야 합니다.
`Dual(WeekYear(Date) & '-W' & Week(Date), WeekStart (Date)) as YearWeek`	이 예에서는 '2012-W22' 형식의 텍스트 값으로 YearWeek 필드를 만들고 동시에 해당 주의 시작 요일의 날짜에 해당하는 숫자 값을 할당합니다 (예: 41057. 스크립트에서 사용하기 위해서는 Date의 값을 로드해야 합니다.

Interval

Interval()은 데이터 로드 스크립트, 운영 체제 또는 서식 문자열(지정된 경우)에서 시스템 변수의 서식을 사용하여 숫자를 시간 간격으로 서식 지정합니다.

간격의 서식은 시간, 일 또는 일, 시간, 분, 초 및 백분초의 조합으로 지정할 수 있습니다.

구문:

```
Interval(number[, format])
```

반환 데이터 유형: dual

인수:

인수

인수	설명
number	서식 지정할 숫자입니다.
format	결과 간격 문자열의 서식을 지정하는 방법을 설명하는 문자열입니다. 생략하는 경우 운영 체제에서 설정한 간단한 날짜 서식, 시간 서식 및 소수점 구분 기호가 사용됩니다.

예 및 결과:
아래 예에서는 다음과 같은 기본 설정이 사용된다고 가정합니다.

- 날짜 서식 설정 1: YY-MM-DD
- 날짜 서식 설정 2: hh:mm:ss
- 숫자 소수점 구분 기호: .

결과 테이블

예	문자열	숫자
`Interval( A )` 여기서 A=0.375	09:00:00	0.375
`Interval( A )` 여기서 A=1.375	33:00:00	1.375
`Interval( A, 'D hh:mm' )` 여기서 A=1.375	1 09:00	1.375
`Interval( A-B, 'D hh:mm' )` 여기서 A=97-08-06 09:00:00 및 B=96-08-06 00:00:00	365 09:00	365.375

Money

Money()는 데이터 로드 스크립트 또는 운영 체제(서식 문자열을 지정하지 않은 경우)에 설정된 시스템 변수에 지정된 서식 및 선택적 소수점 구분 기호와 천 단위 구분 기호를 사용하여 표현식을 숫자 화폐 값으로 지정합니다.

구문:

```
Money(number[, format[, dec_sep[, thou_sep]]])
```

반환 데이터 유형: dual

인수:

인수

인수	설명
number	서식 지정할 숫자입니다.
format	결과 화폐 문자열의 서식을 지정하는 방법을 설명하는 문자열입니다.
dec_sep	소수점 구분 기호를 지정하는 문자열입니다.
thou_sep	천 단위 구분 기호를 지정하는 문자열입니다.

인수 2~4를 생략하면 운영 체제에 설정된 통화 서식이 사용됩니다.

예 및 결과:
아래 예에서는 다음과 같은 기본 설정이 사용된다고 가정합니다.

- MoneyFormat setting 1: kr ##0,00, MoneyThousandSep' '
- MoneyFormat setting 2: $ #,##0.00, MoneyThousandSep','

```
Money( A )
```
여기서 A=35648

결과 테이블

결과	설정 1	설정 2
문자열:	kr 35 648,00	$ 35,648.00
숫자:	35648.00	35648.00

```
Money( A, '#,##0 ¥', '.' , ',' )
```
여기서 A=3564800

결과 테이블

결과	설정 1	설정 2
문자열:	3,564,800 ¥	3,564,800 ¥
숫자:	3564800	3564800

Num

Num()은 숫자의 형식을 지정합니다. 즉, 입력의 숫자 값을 변환하여 두 번째 매개 변수에 지정된 형식을 사용하여 텍스트를 표시합니다. 두 번째 매개 변수를 생략하면 데이터 로드 스크립트에 설정된 소수 및 천 단위 구분 기호를 사용합니다. 사용자 지정 소수점 구분 기호 및 천 단위 구분 기호는 옵션 파라메타입니다.

구문:

```
Num(number[, format[, dec_sep [, thou_sep]]])
```

반환 데이터 유형: dual

Num 함수는 문자열과 숫자 값이 모두 포함된 이중 값을 반환합니다. 이 함수는 입력 표현식의 숫자 값을 사용하여 숫자를 나타내는 문자열을 생성합니다.

인수:

인수

인수	설명
number	서식 지정할 숫자입니다.
format	결과 문자열의 서식을 지정하는 방법을 지정하는 문자열입니다. 생략하면 데이터 로드 스크립트에 설정된 소수 및 천 단위 구분 기호가 사용됩니다.
dec_sep	소수점 구분 기호를 지정하는 문자열입니다. 생략하면 데이터 로드 스크립트에 설정된 변수 DecimalSep의 값이 사용됩니다.
thou_sep	천 단위 구분 기호를 지정하는 문자열입니다. 생략하면 데이터 로드 스크립트에 설정된 변수 ThousandSep의 값이 사용됩니다.

예: 차트 표현식

다음 표는 필드 A가 35648.312일 때의 결과를 보여 줍니다.

결과

A	결과
Num(A)	35648.312(스크립트의 환경 변수에 따라 다름)
Num(A, '0.0', '.')	35648.3
Num(A, '0,00', ',')	35648,31
Num(A, '#,##0.0', '.', ',')	35,648.3
Num(A, '# ##0', ',', ' ')	35 648

예: 로드 스크립트

로드 스크립트

*Num*은 천 단위와 소수 구분 기호가 스크립트에 이미 설정된 경우에도 숫자 서식을 지정하기 위해 로드 스크립트에서 사용할 수 있습니다. 아래의 로드 스크립트에는 특정 천 단위 및 소수 구분 기호가 포함되어 있지만 다른 방식으로 데이터 서식을 지정하기 위해 *Num*을 사용합니다.

데이터 로드 편집기에서 새 섹션을 만든 다음 예제 스크립트를 추가하고 실행합니다. 그런 다음, 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.

```
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
];
```

로드 스크립트에서 *Num* 함수의 다른 용도의 결과를 보여 주는 Qlik Sense 테이블입니다. 테이블의 네 번째 열에는 잘못된 형식이 사용되었습니다(예: 목적).

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

예: 로드 스크립트

로드 스크립트

숫자를 백분율로 서식을 지정하기 위해 로드 스크립트에서 *Num*을 사용할 수 있습니다.

데이터 로드 편집기에서 새 섹션을 만든 다음 예제 스크립트를 추가하고 실행합니다. 그런 다음, 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.

```
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
];
```

백분율로 서식을 지정하기 위해 로드 스크립트에서 사용하는 *Num* 함수의 결과를 보여 주는 Qlik Sense 테이블입니다.

Discount	Discount #,##0%
0.333333333333333	33%
0.22	22%
0	0%
.14	14%
0.1	10%
0	0%
75	7,500%

Time

Time()은 데이터 로드 스크립트 또는 운영 체제(서식 문자열을 지정하지 않은 경우)의 시스템 변수에 설정된 시간 서식을 사용하여 표현식을 시간 값으로 서식 지정합니다.

구문:

```
Time(number[, format])
```

반환 데이터 유형: dual

인수:

인수

인수	설명
number	서식 지정할 숫자입니다.
format	결과 시간 문자열의 서식을 지정하는 방법을 설명하는 문자열입니다. 생략하는 경우 운영 체제에서 설정한 간단한 날짜 서식, 시간 서식 및 소수점 구분 기호가 사용됩니다.

예 및 결과:
아래 예에서는 다음과 같은 기본 설정이 사용된다고 가정합니다.

- 시간 서식 설정 1: hh:mm:ss
- 시간 서식 설정 2: hh.mm.ss

```
Time( A )
```
여기서 A=0.375

결과 테이블

결과	설정 1	설정 2
문자열:	09:00:00	09.00.00
숫자:	0.375	0.375

```
Time( A )
```
여기서 A=35648.375

결과 테이블

결과	설정 1	설정 2
문자열:	09:00:00	09.00.00
숫자:	35648.375	35648.375

```
Time( A, 'hh-mm' )
```
여기서 A=0.99999

결과 테이블

결과	설정 1	설정 2
문자열:	23-59	23-59
숫자:	0.99999	0.99999

Timestamp

TimeStamp()는 데이터 로드 스크립트 또는 운영 체제(서식 문자열을 지정하지 않은 경우)의 시스템 변수에 설정된 타임스탬프 서식을 사용하여 표현식을 날짜 및 시간 값으로 서식 지정합니다.

구문:

```
Timestamp(number[, format])
```

반환 데이터 유형: dual

인수:

인수

인수	설명
number	서식 지정할 숫자입니다.
format	결과 타임스탬프 문자열의 서식을 지정하는 방법을 설명하는 문자열입니다. 생략하는 경우 운영 체제에서 설정한 간단한 날짜 서식, 시간 서식 및 소수점 구분 기호가 사용됩니다.

예 및 결과:
아래 예에서는 다음과 같은 기본 설정이 사용된다고 가정합니다.

- 타임스탬프 서식 설정 1: YY-MM-DD hh:mm:ss
- 타임스탬프 서식 설정 2: M/D/YY hh:mm:ss

```
Timestamp( A )
```
여기서 A=35648.375

결과 테이블

결과	설정 1	설정 2
문자열:	97-08-06 09:00:00	8/6/97 09:00:00
숫자:	35648.375	35648.375

```
Timestamp( A,'YYYY-MM-DD hh.mm')
```
여기서 A=35648

결과 테이블

결과	설정 1	설정 2
문자열:	1997-08-06 00.00	1997-08-06 00.00
숫자:	35648	35648

8.13 일반 숫자 함수

일반 숫자 함수에서 인수는 **x**를 실수로 해석해야 하는 표현식입니다. 데이터 로드 스크립트와 차트 표현식 모두에서 모든 함수를 사용할 수 있습니다.

일반 숫자 함수 개요

각 함수는 개요가 끝난 후에 더 자세히 설명합니다. 구문에서 함수 이름을 클릭하여 해당 함수에 대한 상세 설명에 즉시 액세스할 수도 있습니다.

bitcount

BitCount()는 10진수와 동등한 2진수에서 1로 설정된 비트 수를 반환합니다. 즉, 이 함수는 **integer_number**에서 설정된 비트 수를 반환하며, 여기서 **integer_number**는 부호 있는 32비트 정수로 해석됩니다.

```
BitCount(integer_number)
```

div

Div()는 첫 번째 인수를 두 번째 인수로 산술적으로 나눈 정수 부분을 반환합니다. 두 파라메타는 모두 실수로 해석되므로 정수일 필요가 없습니다.

```
Div (integer_number1, integer_number2)
```

fabs

Fabs()는 **x**의 절대값을 반환합니다. 결과는 양수입니다.

```
Fabs (x)
```

fact

Fact()는 양의 정수 **x**의 계승을 반환합니다.

```
Fact (x)
```

frac

Frac()는 **x**의 분위수 부분을 반환합니다.

```
Frac (x)
```

sign

Sign()은 **x**가 양수, 0 또는 음수인가에 따라 1, 0 또는 -1을 반환합니다.

```
Sign (x)
```

조합 및 치환 함수

combin

Combin()은 **p** 항목 집합에서 선택할 수 있는 **q** 요소의 조합 수를 반환합니다. 다음과 같은 식으로 표현됩니다. `Combin(p,q) = p! / q!(p-q)!` 항목이 선택되는 순서는 중요하지 않습니다.

```
Combin (p, q)
```

permut

Permut()는 **p** 항목 집합에서 선택할 수 있는 **q** 요소의 순열 수를 반환합니다. 다음과 같은 식으로 표현됩니다. `Permut(p,q) = (p)! / (p - q)!` 항목이 선택되는 순서는 중요하지 않습니다.

```
Permut (p, q)
```

Modulo 함수

fmod

fmod()는 첫 번째 인수(피제수)를 두 번째 인수(제수)로 나눈 정수의 나머지 부분을 반환하는 일반화된 모듈 함수입니다. 결과는 실수입니다. 두 인수는 모두 실수로 해석되므로 정수일 필요가 없습니다.

```
Fmod (a, b)
```

mod

Mod()는 정수 나누기의 음수가 아닌 나머지를 반환하는 수학적 모듈 함수입니다. 첫 번째 인수는 피제수이고 두 번째 인수는 제수이며, 두 인수는 모두 정수 값이어야 합니다.

```
Mod (integer_number1, integer_number2)
```

패리티 함수

even

Even()은 **integer_number**가 짝수 정수 또는 0일 경우 True(-1)를 반환합니다. **integer_number**가 홀수 정수일 경우는 False(0)를, **integer_number**가 정수가 아닐 경우는 NULL을 반환합니다.

```
Even (integer_number)
```

odd

Odd()는 **integer_number**가 홀수 정수 또는 0일 경우 True(-1)를 반환합니다. **integer_number**가 짝수 정수일 경우는 False(0)를, **integer_number**가 정수가 아닐 경우는 NULL을 반환합니다.

```
Odd (integer_number)
```

절사 함수

ceil

Ceil()은 **offset** 숫자에 의해 시프트된 **step**의 가장 가까운 배수로 숫자를 올림합니다.

```
Ceil (x[, step[, offset]])
```

floor

Floor()는 **offset** 숫자에 의해 시프트된 **step**의 가장 가까운 배수로 숫자를 내림합니다.

```
Floor (x[, step[, offset]])
```

round

Round()는 **offset** 숫자에 의해 시프트된 **step**의 가장 가까운 배수로 숫자를 올림 또는 내림한 결과를 반환합니다.

```
Round ( x [ , step [ , offset ]])
```

BitCount

BitCount()는 10진수와 동등한 2진수에서 1로 설정된 비트 수를 반환합니다. 즉, 이 함수는 **integer_number**에서 설정된 비트 수를 반환하며, 여기서 **integer_number**는 부호 있는 32비트 정수로 해석됩니다.

구문:

```
BitCount(integer_number)
```

반환 데이터 유형: 정수

예 및 결과:

예 및 결과

예	결과
BitCount (3)	3은 이진수 11이므로 2를 반환합니다.
BitCount (-1)	-1은 이진수로 1이 64개이므로 이는 64를 반환합니다.

Ceil

Ceil()은 **offset** 숫자에 의해 시프트된 **step**의 가장 가까운 배수로 숫자를 올림합니다.

입력 숫자를 반내림하는 **floor** 함수와 비교합니다.

구문:

```
Ceil(x[, step[, offset]])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
x	입력 숫자입니다.
step	간격 증가값입니다. 기본값은 1입니다.
offset	단계 간격의 기수를 정의합니다. 기본값은 0입니다.

예 및 결과:

예 및 결과

예	결과
`Ceil(2.4 )`	3을 반환합니다. 이 예에서, 단계의 크기는 1이고 단계 간격의 기수는 0입니다. 간격은 ...0 < x <=1, 1 < x <= 2, **2< x <=3**, 3< x <=4...입니다.
`Ceil(4.2 )`	5를 반환합니다.
`Ceil(3.88 ,0.1)`	3.9를 반환합니다. 이 예에서, 간격의 크기는 0.1이고 간격의 기수는 0입니다. 간격은 ... 3.7 < x <= 3.8, **3.8 < x <= 3.9**, 3.9 < x <= 4.0...입니다.
`Ceil(3.88 ,5)`	5를 반환합니다.
`Ceil(1.1 ,1)`	2를 반환합니다.
`Ceil(1.1 ,1,0.5)`	1.5를 반환합니다. 이 예에서, 단계의 크기는 1이고 오프셋은 0.5입니다. 즉, 단계 간격의 기수는 0.5이며 0이 아닙니다. 간격은 ...**0.5 < x <=1.5**, 1.5 < x <= 2.5, 2.5< x <=3.5, 3.5< x <=4.5...입니다.
`Ceil(1.1 ,1,-0.01)`	1.99를 반환합니다. 간격은 ...-0.01< x <= 0.99, **0.99< x <= 1.99**, 1.99 < x <=2.99...입니다.

Combin

Combin()은 **p** 항목 집합에서 선택할 수 있는 **q** 요소의 조합 수를 반환합니다. 다음과 같은 식으로 표현됩니다. `Combin(p,q) = p! / q!(p-q)!` 항목이 선택되는 순서는 중요하지 않습니다.

구문:

```
Combin(p, q)
```

반환 데이터 유형: 정수

제한 사항:

정수가 아닌 항목은 절사됩니다.

예 및 결과:

예 및 결과

예	결과
총 35개의 로또 번호 중에서 숫자 7개의 조합을 얼마나 많이 선택할 수 있습니까? `Combin( 35,7 )`	6,724,520을 반환합니다.

Div

Div()는 첫 번째 인수를 두 번째 인수로 산술적으로 나눈 정수 부분을 반환합니다. 두 파라메타는 모두 실수로 해석되므로 정수일 필요가 없습니다.

구문:

```
Div(integer_number1, integer_number2)
```

반환 데이터 유형: 정수

예 및 결과:

예 및 결과

예	결과
`Div( 7,2 )`	3을 반환합니다.
`Div( 7.1,2.3 )`	3을 반환합니다.
`Div( 9,3 )`	3을 반환합니다.
`Div( -4,3 )`	-1을 반환합니다.
`Div( 4,-3 )`	-1을 반환합니다.
`Div( -4,-3 )`	1을 반환합니다.

Even

Even()은 **integer_number**가 짝수 정수 또는 0일 경우 True(-1)를 반환합니다. **integer_number**가 홀수 정수일 경우는 False(0)를, **integer_number**가 정수가 아닐 경우는 NULL을 반환합니다.

구문:

```
Even(integer_number)
```

반환 데이터 유형: 부울

예 및 결과:

예 및 결과

예	결과
`Even( 3 )`	0, False를 반환합니다.
`Even( 2 * 10 )`	-1, True를 반환합니다.
`Even( 3.14 )`	NULL를 반환합니다.

Fabs

Fabs()는 **x**의 절대값을 반환합니다. 결과는 양수입니다.

구문:

```
fabs(x)
```

반환 데이터 유형: 숫자

예 및 결과:

예 및 결과

예	결과
`fabs( 2.4 )`	2.4를 반환합니다.
`fabs( -3.8 )`	3.8을 반환합니다.

Fact

Fact()는 양의 정수 **x**의 계승을 반환합니다.

구문:

```
Fact(x)
```

반환 데이터 유형: 정수

제한 사항:

숫자 **x**가 정수가 아닐 경우 절사됩니다. 양수가 아니면 NULL이 반환됩니다.

예 및 결과:

예 및 결과

예	결과
`Fact( 1 )`	1을 반환합니다.
`Fact( 5 )`	120(1 * 2 * 3 * 4 * 5 = 120)을 반환합니다.
`Fact( -5 )`	NULL를 반환합니다.

Floor

Floor()는 **offset** 숫자에 의해 시프트된 **step**의 가장 가까운 배수로 숫자를 내림합니다.

입력 숫자를 반올림하는 **ceil** 함수와 비교합니다.

구문:

```
Floor(x[, step[, offset]])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
x	입력 숫자입니다.
step	간격 증가값입니다. 기본값은 1입니다.
offset	단계 간격의 기수를 정의합니다. 기본값은 0입니다.

예 및 결과:

예 및 결과

예	결과
`Floor(2.4)`	2를 반환합니다. 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)`	4를 반환합니다.
`Floor(3.88 ,0.1)`	3.8을 반환합니다. 이 예에서, 간격의 크기는 0.1이고 간격의 기수는 0입니다. 간격은 ... 3.7 <= x < 3.8, **3.8 <= x < 3.9**, 3.9 <= x < 4.0... 입니다.

예	결과
`Floor(3.88 ,5)`	0을 반환합니다.
`Floor(1.1 ,1)`	1을 반환합니다.
`Floor(1.1 ,1,0.5)`	0.5를 반환합니다. 이 예에서, 단계의 크기는 1이고 오프셋은 0.5입니다. 즉, 단계 간격의 기수는 0.5이며 0이 아닙니다. 간격은 ...**0.5 <= x <1.5**, 1.5 <= x < 2.5, 2.5<= x <3.5,... 입니다.

Fmod

fmod()는 첫 번째 인수(피제수)를 두 번째 인수(제수)로 나눈 정수의 나머지 부분을 반환하는 일반화된 모듈 함수입니다. 결과는 실수입니다. 두 인수는 모두 실수로 해석되므로 정수일 필요가 없습니다.

구문:

```
fmod(a, b)
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
a	피제수
b	제수

예 및 결과:

예 및 결과

예	결과
`fmod( 7,2 )`	1을 반환합니다.
`fmod( 7.5,2 )`	1.5를 반환합니다.
`fmod( 9,3 )`	0을 반환합니다.
`fmod( -4,3 )`	-1을 반환합니다.
`fmod( 4,-3 )`	1을 반환합니다.
`fmod( -4,-3 )`	-1을 반환합니다.

Frac

Frac()는 **x**의 분위수 부분을 반환합니다.

분위수는 Frac(x) + Floor(x) = x와 같이 정의됩니다. 간단히 말해서, 이는 양수의 분수 부분이 숫자(x)와 분수 부분 앞에 선행하는 정수 사이의 차이임을 의미합니다.

예: 11.43의 분수 부분 = 11.43 - 11 = 0.43

-1.4와 같은 음수의 경우, Floor(-1.4) = -2이며 이는 다음과 같은 결과를 산출합니다.

-1.4의 분수 부분 = -1.4 - (-2) = -1.4 + 2 = 0.6

구문:

```
Frac(x)
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
x	분위수를 반환할 숫자입니다.

예 및 결과:

예 및 결과

예	결과
Frac(11.43)	0.43을 반환합니다.
Frac(-1.4)	0.6을 반환합니다.
타임스탬프의 숫자 표현에서 시간 구성 요소를 추출하여 날짜를 생략합니다. Time(Frac(44518.663888889))	오후 3:56:00 반환

Mod

Mod()는 정수 나누기의 음수가 아닌 나머지를 반환하는 수학적 모듈 함수입니다. 첫 번째 인수는 피제수이고 두 번째 인수는 제수이며, 두 인수는 모두 정수 값이어야 합니다.

구문:

```
Mod(integer_number1, integer_number2)
```

반환 데이터 유형: 정수

제한 사항:

integer_number2는 0보다 커야 합니다.

예 및 결과:

예 및 결과

예	결과
`Mod( 7,2 )`	1을 반환합니다.
`Mod( 7.5,2 )`	NULL를 반환합니다.
`Mod( 9,3 )`	0을 반환합니다.
`Mod( -4,3 )`	2를 반환합니다.
`Mod( 4,-3 )`	NULL를 반환합니다.
`Mod( -4,-3 )`	NULL를 반환합니다.

Odd

Odd()는 **integer_number**가 홀수 정수 또는 0일 경우 True(-1)를 반환합니다. **integer_number**가 짝수 정수일 경우는 False(0)를, **integer_number**가 정수가 아닐 경우는 NULL을 반환합니다.

구문:

```
Odd(integer_number)
```

반환 데이터 유형: 부울

예 및 결과:

예 및 결과

예	결과
`Odd( 3 )`	-1, True를 반환합니다.
`Odd( 2 * 10 )`	0, False를 반환합니다.
`Odd( 3.14 )`	NULL를 반환합니다.

Permut

Permut()는 **p** 항목 집합에서 선택할 수 있는 **q** 요소의 순열 수를 반환합니다. 다음과 같은 식으로 표현됩니다. `Permut(p,q) = (p)! / (p - q)!` 항목이 선택되는 순서는 중요하지 않습니다.

구문:

```
Permut(p, q)
```

반환 데이터 유형: 정수

제한 사항:

정수가 아닌 인수는 절사됩니다.

예 및 결과:

예 및 결과

예	결과
8명이 참가한 100 m 결승 후에 금, 은, 동메달을 나눠줄 수 있는 방법은 몇 가지입니까? `Permut( 8,3 )`	336을 반환합니다.

Round

Round()는 **offset** 숫자에 의해 시프트된 **step**의 가장 가까운 배수로 숫자를 올림 또는 내림한 결과를 반환합니다.

절사할 숫자가 정확히 간격의 중간일 경우 반올림됩니다.

구문:

```
Round(x[, step[, offset]])
```

반환 데이터 유형: 숫자

> *부동 소수점 숫자를 절사하면 잘못된 결과가 나올 수 있습니다. 이러한 절사 오차는 부동 소수점 숫자가 한정된 수의 이진 숫자로 표현되기 때문에 발생합니다. 따라서 이미 절사된 숫자를 사용하여 계산한 결과가 나옵니다. 이러한 절사 오차가 작업에 영향을 미치는 경우에는 절사 전에 숫자를 곱하여 정수로 변환합니다.*

인수:

인수

인수	설명
x	입력 숫자입니다.
step	간격 증가값입니다. 기본값은 1입니다.
offset	단계 간격의 기수를 정의합니다. 기본값은 0입니다.

예 및 결과:

예 및 결과

예	결과
`Round(3.8 )`	4를 반환합니다. 이 예에서, 단계의 크기는 1이고 단계 간격의 기수는 0입니다. 간격은 ...0 <= x <1, 1 <= x < 2, 2<= x <3, **3<= x <4**...입니다.
`Round(3.8,4 )`	4를 반환합니다.
`Round(2.5 )`	3을 반환합니다. 이 예에서, 단계의 크기는 1이고 단계 간격의 기수는 0입니다. 간격은 ...0 <= x <1, 1 <= x <2, **2<= x <3**...입니다.
`Round(2,4 )`	4를 반환합니다. 2는 단계 간격 4의 정확히 절반이므로 반올림됩니다. 이 예에서, 단계의 크기는 4이고 단계 간격의 기수는 0입니다. 간격은 ...**0 <= x <4**, 4 <= x <8, 8<= x <12...입니다.
`Round(2,6 )`	0을 반환합니다. 2는 단계 간격 6의 절반보다 작으므로 반내림됩니다. 이 예에서, 단계의 크기는 6이고 단계 간격의 기수는 0입니다. 간격은 ...**0 <= x <6**, 6 <= x <12, 12<= x <18...입니다.
`Round(3.88 ,0.1)`	3.9를 반환합니다. 이 예에서, 단계의 크기는 0.1이고 단계 간격의 기수는 0입니다. 간격은 ... 3.7 <= x <3.8, **3.8 <= x <3.9**, 3.9 <= x < 4.0...입니다.
`Round (3.88875,1/1000)`	3.889를 반환합니다. 이 예에서 단계의 크기는 0.001이며 숫자를 반올림하고 소수점 이하 세 자리로 제한합니다.
`Round(3.88 ,5)`	5를 반환합니다.
`Round(1.1 ,1,0.5)`	1.5를 반환합니다. 이 예에서, 단계의 크기는 1이고 단계 간격의 기수는 0.5입니다. 간격은 ...**0.5 <= x <1.5**, 1.5 <= x <2.5, 2.5<= x <3.5...입니다.

Sign

Sign()은 **x**가 양수, 0 또는 음수인가에 따라 1, 0 또는 -1을 반환합니다.

구문:

```
Sign(x)
```

반환 데이터 유형: 숫자

제한 사항:

숫자 값이 발견되지 않으면 NULL이 반환됩니다.

예 및 결과:

예 및 결과

예	결과
`Sign( 66 )`	1을 반환합니다.
`Sign( 0 )`	0을 반환합니다.
`Sign( - 234 )`	-1을 반환합니다.

8.14 특정 지역 관련 함수

이 함수는 맵 시각화에서 특정 지역 관련 데이터를 처리하는 데 사용됩니다. Qlik Sense는 특정 지역 관련 데이터에 대한 GeoJSON 사양을 따르고 다음을 지원합니다.

- Point
- Linestring
- Polygon
- Multipolygon

GeoJSON 사양에 대한 자세한 내용은 다음을 참조하십시오.
GeoJSON.org

특정 지역 관련 함수 개요

각 함수는 개요가 끝난 후에 더 자세히 설명합니다. 구문에서 함수 이름을 클릭하여 해당 함수에 대한 상세 설명에 즉시 액세스할 수도 있습니다.

특정 지역 관련 함수에는 집계와 비집계의 두 가지 범주가 있습니다.

집계 함수는 입력으로 도형 집합(포인트 또는 영역)을 사용하며 단일 도형을 반환합니다. 예를 들어, 여러 영역을 병합할 수 있으며 해당 집계에 대한 단일 경계를 맵에 그릴 수 있습니다.

비집계 함수는 단일 도형을 사용하여 하나의 도형을 반환합니다. 예를 들어 GeoGetPolygonCenter() 함수의 경우, 한 영역의 경계 도형이 입력으로 설정되면 해당 영역의 중심에 대한 포인트 도형(경도 및 위도)이 반환됩니다.

다음은 집계 함수입니다.

GeoAggrGeometry

GeoAggrGeometry()는 여러 영역을 더 큰 영역으로 집계하는 데 사용됩니다(예: 여러 하위 지역을 지역으로 집계).

```
GeoAggrGeometry (field_name)
```

GeoBoundingBox

GeoBoundingBox()는 도형을 영역으로 집계하고 모든 좌표가 포함된 가장 작은 경계 상자를 계산하는 데 사용됩니다.

```
GeoBoundingBox (field_name)
```

GeoCountVertex

GeoCountVertex()는 다각형에 포함된 꼭지점의 수를 찾는 데 사용됩니다.

```
GeoCountVertex(field_name)
```

GeoInvProjectGeometry

GeoInvProjectGeometry()는 도형을 영역으로 집계하고 역투사를 적용하는 데 사용됩니다.

```
GeoInvProjectGeometry(type, field_name)
```

GeoProjectGeometry

GeoProjectGeometry()는 도형을 영역으로 집계하고 투사를 적용하는 데 사용됩니다.

```
GeoProjectGeometry(type, field_name)
```

GeoReduceGeometry

GeoReduceGeometry()를 사용하면 도형의 꼭지점 수를 줄이고 여러 영역을 하나의 영역으로 집계하지만, 여전히 개별 영역에서 경계선을 표시합니다.

```
GeoReduceGeometry (geometry)
```

다음은 비집계 함수입니다.

GeoGetBoundingBox

GeoGetBoundingBox()는 스크립트 및 차트 표현식에서 도형의 모든 좌표가 포함된 가장 작은 특정 지역 관련 경계 상자를 계산하는 데 사용됩니다.

```
GeoGetBoundingBox (geometry)
```

GeoGetPolygonCenter

GeoGetPolygonCenter()는 스크립트 및 차트 표현식에서 도형의 중심점을 계산하여 반환하는 데 사용됩니다.

```
GeoGetPolygonCenter (geometry)
```

GeoMakePoint

GeoMakePoint()는 스크립트 및 차트 표현식에서 위도와 경도가 있는 포인트를 만들고 태그 지정하는 데 사용됩니다.

```
GeoMakePoint (lat_field_name, lon_field_name)
```

GeoProject

GeoProject()는 스크립트 및 차트 표현식에서 도형에 투사를 적용하는 데 사용됩니다.

```
GeoProject (type, field_name)
```

GeoAggrGeometry

GeoAggrGeometry()는 여러 영역을 더 큰 영역으로 집계하는 데 사용됩니다(예: 여러 하위 지역을 지역으로 집계).

구문:

```
GeoAggrGeometry(field_name)
```

반환 데이터 유형: 문자열

인수:

인수

인수	설명
field_name	표시할 도형이 포함된 필드를 나타내는 필드 또는 표현식입니다. 위도와 경도를 제공하는 포인트(또는 포인트의 집합)나 영역이 될 수 있습니다.

일반적으로 **GeoAggrGeometry()**는 특정 지역 관련 경계 데이터를 결합하는 데 사용될 수 있습니다. 예를 들어, 한 도시의 교외 지역에 대한 우편 번호 영역과 각 영역에 대한 판매 수익이 있을 수 있습니다. 영업직원의 담당 지역에 여러 우편번호 영역이 포함되는 경우 개별 영역보다는 판매 담당 지역별 총 판매량을 나타내고 색칠된 맵에 결과를 표시하는 것이 유용할 수 있습니다.

GeoAggrGeometry()는 개별 교외 지역의 집계를 계산하고 병합된 담당 지역을 데이터 모델에 생성할 수 있습니다. 그러면 판매 담당 지역 경계가 조정되어 데이터를 다시 로드하면 새로 병합된 경계와 수익이 맵에 반영됩니다.

GeoAggrGeometry()는 집계 함수이므로 스크립트에서 사용할 경우 **Group by** 절이 포함된 **LOAD** 문이 필요합니다.

***GeoAggrGeometry()**를 사용하여 생성된 맵의 경계선은 병합된 영역의 경계선에 해당합니다. 사전 집계된 영역의 개별 경계선을 표시하려면 **GeoReduceGeometry()**를 사용합니다.*

예:

이 예에서는 영역 데이터가 포함된 KML 파일을 로드한 다음 집계된 영역 데이터가 있는 테이블을 로드합니다.

```
[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()는 도형을 영역으로 집계하고 모든 좌표가 포함된 가장 작은 경계 상자를 계산하는 데 사용됩니다.

GeoBoundingBox는 왼쪽, 오른쪽, 위쪽, 아래쪽의 네 개의 값 목록으로 나타냅니다.

구문:

```
GeoBoundingBox(field_name)
```

반환 데이터 유형: 문자열

인수:

인수

인수	설명
field_name	표시할 도형이 포함된 필드를 나타내는 필드 또는 표현식입니다. 위도와 경도를 제공하는 포인트(또는 포인트의 집합)나 영역이 될 수 있습니다.

GeoBoundingBox()는 일련의 도형을 집계하고 해당 집계된 도형의 모든 좌표가 포함된 가장 작은 사각형에 대한 네 좌표를 반환합니다.

결과를 맵에 시각화하려면 네 좌표의 결과 문자열을 다각형 형식으로 변환하고, 변환된 필드에 지리 다각형(geopolygon) 형식을 사용하여 태그를 지정한 후 해당 필드를 맵 개체에 끌어서 놓습니다. 그러면 맵 시각화에 사각형 상자가 표시됩니다.

GeoCountVertex

GeoCountVertex()는 다각형에 포함된 꼭지점의 수를 찾는 데 사용됩니다.

구문:

```
GeoCountVertex(field_name)
```

반환 데이터 유형: 정수

인수:

인수

인수	설명
field_name	표시할 도형이 포함된 필드를 나타내는 필드 또는 표현식입니다. 위도와 경도를 제공하는 포인트(또는 포인트의 집합)나 영역이 될 수 있습니다.

GeoGetBoundingBox

GeoGetBoundingBox()는 스크립트 및 차트 표현식에서 도형의 모든 좌표가 포함된 가장 작은 특정 지역 관련 경계 상자를 계산하는 데 사용됩니다.

GeoBoundingBox() 함수를 사용하여 생성된 특정 지역 관련 경계 상자는 왼쪽, 오른쪽, 위쪽, 아래쪽의 네 개의 값 목록으로 나타냅니다.

구문:

```
GeoGetBoundingBox(field_name)
```

반환 데이터 유형: 문자열

인수:

인수

인수	설명
field_name	표시할 도형이 포함된 필드를 나타내는 필드 또는 표현식입니다. 위도와 경도를 제공하는 포인트(또는 포인트의 집합)나 영역이 될 수 있습니다.

로드 시 오류가 발생하므로 데이터 로드 편집기에서 이 함수 및 기타 특정 지역 관련 비집계 함수와 함께 ***Group by*** *절을 사용하지 마십시오.*

GeoGetPolygonCenter

GeoGetPolygonCenter()는 스크립트 및 차트 표현식에서 도형의 중심점을 계산하여 반환하는 데 사용됩니다.

경우에 따라서는 맵을 색칠하는 대신 점을 표시해야 합니다. 기존 특정 지역 데이터를 영역 도형 형식으로만 사용할 수 있는 경우(예: 경계선)에는 **GeoGetPolygonCenter()**를 사용하여 영역의 중심에 대한 경도와 위도 쌍을 검색합니다.

구문:

```
GeoGetPolygonCenter(field_name)
```

반환 데이터 유형: 문자열

인수:

인수

인수	설명
field_name	표시할 도형이 포함된 필드를 나타내는 필드 또는 표현식입니다. 위도와 경도를 제공하는 포인트(또는 포인트의 집합)나 영역이 될 수 있습니다.

로드 시 오류가 발생하므로 데이터 로드 편집기에서 이 함수 및 기타 특정 지역 관련 비집계 함수와 함께 ***Group by*** *절을 사용하지 마십시오.*

GeoInvProjectGeometry

GeoInvProjectGeometry()는 도형을 영역으로 집계하고 역투사를 적용하는 데 사용됩니다.

구문:

```
GeoInvProjectGeometry(type, field_name)
```

반환 데이터 유형: 문자열

인수:

인수

인수	설명
type	맵의 도형을 변환하는 데 사용되는 투사 유형입니다. 1:1 투사를 나타내는 'unit'(기본값) 또는 표준 메르카토르 투사법을 사용하는 'mercator' 중 하나를 선택할 수 있습니다.
field_name	표시할 도형이 포함된 필드를 나타내는 필드 또는 표현식입니다. 위도와 경도를 제공하는 포인트(또는 포인트의 집합)나 영역이 될 수 있습니다.

예:

스크립팅 예

예	결과
Load 문에서: `GeoInvProjectGeometry ('mercator',AreaPolygon) as InvProjectGeometry`	**AreaPolygon**으로 로드된 도형은 메르카토르 투사의 역변환을 사용하여 변환되며 시각화에서 사용하기 위해 **InvProjectGeometry**로 저장됩니다.

GeoMakePoint

GeoMakePoint()는 스크립트 및 차트 표현식에서 위도와 경도가 있는 포인트를 만들고 태그 지정하는 데 사용됩니다. GeoMakePoint는 경도와 위도의 순서로 점을 반환합니다.

구문:

```
GeoMakePoint(lat_field_name, lon_field_name)
```

반환 데이터 유형: string, formatted [longitude, latitude]

인수:

인수

인수	설명
lat_field_name	포인트의 위도를 나타내는 필드 또는 필드를 참조하는 표현식입니다.
lon_field_name	포인트의 경도를 나타내는 필드 또는 필드를 참조하는 표현식입니다.

로드 시 오류가 발생하므로 데이터 로드 편집기에서 이 함수 및 기타 특정 지역 관련 비집계 함수와 함께 ***Group by*** *절을 사용하지 마십시오.*

GeoProject

GeoProject()는 스크립트 및 차트 표현식에서 도형에 투사를 적용하는 데 사용됩니다.

구문:

```
GeoProject(type, field_name)
```

반환 데이터 유형: 문자열

인수:

인수

인수	설명
type	맵의 도형을 변환하는 데 사용되는 투사 유형입니다. 1:1 투사를 나타내는 'unit'(기본값) 또는 웹 메르카토르 투사법을 사용하는 'mercator' 중 하나를 선택할 수 있습니다.
field_name	표시할 도형이 포함된 필드를 나타내는 필드 또는 표현식입니다. 위도와 경도를 제공하는 포인트(또는 포인트의 집합)나 영역이 될 수 있습니다.

로드 시 오류가 발생하므로 데이터 로드 편집기에서 이 함수 및 기타 특정 지역 관련 비집계 함수와 함께 ***Group by*** *절을 사용하지 마십시오.*

예:

스크립트 예

예	결과
Load 문에서: `GeoProject('mercator',Area) as GetProject`	메르카토르 투사법이 **Area**로 로드된 도형에 적용되며, 결과는 **GetProject**로 저장됩니다.

GeoProjectGeometry

GeoProjectGeometry()는 도형을 영역으로 집계하고 투사를 적용하는 데 사용됩니다.

구문:

```
GeoProjectGeometry(type, field_name)
```

반환 데이터 유형: 문자열

인수:

인수

인수	설명
type	맵의 도형을 변환하는 데 사용되는 투사 유형입니다. 1:1 투사를 나타내는 'unit'(기본값) 또는 웹 메르카토르 투사법을 사용하는 'mercator' 중 하나를 선택할 수 있습니다.
field_name	표시할 도형이 포함된 필드를 나타내는 필드 또는 표현식입니다. 위도와 경도를 제공하는 포인트(또는 포인트의 집합)나 영역이 될 수 있습니다.

예:

예	결과
Load 문에서: `GeoProjectGeometry ('mercator',AreaPolygon) as ProjectGeometry`	**AreaPolygon**으로 로드된 도형은 메르카토르 투사를 사용하여 변환되며 시각화에서 사용하기 위해 **ProjectGeometry**로 저장됩니다.

GeoReduceGeometry

GeoReduceGeometry()를 사용하면 도형의 꼭지점 수를 줄이고 여러 영역을 하나의 영역으로 집계하지만, 여전히 개별 영역에서 경계선을 표시합니다.

구문:

```
GeoReduceGeometry(field_name[, value])
```

반환 데이터 유형: 문자열

인수:

인수

인수	설명
field_name	표시할 도형이 포함된 필드를 나타내는 필드 또는 표현식입니다. 위도와 경도를 제공하는 포인트(또는 포인트의 집합)나 영역이 될 수 있습니다.
value	도형에 적용하기 위한 축소 양입니다. 범위는 0 ~ 1이며, 0은 축소하지 않음을 나타내고 1은 꼭지점의 최대 축소를 나타냅니다. *복잡한 데이터 셋에 0.9 이상의 value를 사용하면 시각적 표시가 부정확한 수준으로 꼭지점 수를 줄일 수 있습니다.*

또한 **GeoReduceGeometry()**는 **GeoAggrGeometry()**에서 여러 영역을 하나의 영역으로 집계하는 것과 유사한 기능을 수행합니다. **GeoReduceGeometry()**를 사용하면 사전 집계 데이터에서 해당 개별 경계선을 만들 때 나타나는 차이가 맵에 표시됩니다.

GeoReduceGeometry()는 집계 함수이므로 스크립트에서 사용할 경우 **Group by** 절이 포함된 **LOAD** 문이 필요합니다.

예:
이 예에서는 영역 데이터가 포함된 KML 파일을 로드한 다음 축소되고 집계된 영역 데이터가 있는 테이블을 로드합니다.

```
[MapSource]:
LOAD [world.Name],
       [world.Point],
       [world.Area]
 FROM [lib://Downloads/world.kml]
(kml, Table is [World.shp/Features]);

Map:
LOAD world.Name,
     GeoReduceGeometry(world.Area,0.5) as [ReducedArea]
resident MapSource Group By world.Name;

Drop Table MapSource;
```

8.15 해석 함수

해석 함수는 입력 텍스트 필드 또는 표현식의 내용을 평가하고 지정된 날짜 서식을 결과 숫자 값에 적용합니다. 이러한 함수를 사용하면 소수점 구분 기호, 천 단위 구분 기호 및 날짜 서식 등의 특성을 비롯한 데이터 형식에 따라 숫자의 서식을 지정할 수 있습니다.

해석 함수는 문자열과 숫자 값이 모두 포함된 이중 값을 반환하지만 문자열-숫자 변환을 수행하는 것과 유사하게 생각할 수 있습니다. 이 함수는 입력 표현식의 텍스트 값을 사용하여 문자열을 나타내는 숫자를 생성합니다.

반면, 서식 지정 함수는 반대로 작업을 수행합니다. 결과 텍스트의 표시 형식을 지정하는 숫자 표현식을 문자열로 평가합니다.

해석 함수를 사용하지 않을 경우 Qlik Sense가 스크립트 변수 및 운영 체제에서 정의한 숫자 서식, 날짜 서식, 시간 서식의 기본 설정을 사용하여 데이터를 숫자, 날짜, 시간, 타임스탬프 및 문자열의 혼합으로 해석합니다.

모든 해석 함수는 데이터 로드 스크립트와 차트 표현식 모두에서 사용할 수 있습니다.

모든 숫자 표현에는 소수점 구분 기호로 소수점이 사용됩니다.

해석 함수 개요

각 함수는 개요가 끝난 후에 더 자세히 설명합니다. 구문에서 함수 이름을 클릭하여 해당 함수에 대한 상세 설명에 즉시 액세스할 수도 있습니다.

Date#

Date#는 두 번째 인수(지정된 경우)로 지정한 서식을 사용하여 표현식을 날짜로 평가합니다. format-code를 생략하면 운영 체제에 설정된 기본 날짜 서식이 사용됩니다.

```
Date# (page 1219)(text[, format])
```

Interval#

Interval#()은 운영 체제에서 기본 설정된 서식 또는 두 번째 인수(지정된 경우)로 지정된 서식을 사용하여 텍스트 표현식을 시간 간격으로 평가합니다.

```
Interval# (page 1220)(text[, format])
```

Money#

Money#()는 로드 스크립트 또는 운영 체제(서식 문자열을 지정하지 않은 경우)에 설정된 서식을 사용하여 텍스트 문자열을 화폐 값으로 변환합니다. 사용자 지정 소수점 구분 기호 및 천 단위 구분 기호는 옵션 파라메타입니다.

```
Money# (page 1220)(text[, format[, dec_sep[, thou_sep ] ] ])
```

Num#

Num#()은 텍스트 문자열을 숫자 값으로 해석합니다. 즉, 두 번째 매개 변수에 지정된 형식을 사용하여 입력 문자열을 숫자로 변환합니다. 두 번째 매개 변수를 생략하면 데이터 로드 스크립트에 설정된 소수 및 천 단위 구분 기호를 사용합니다. 사용자 지정 소수점 구분 기호 및 천 단위 구분 기호는 옵션 파라메타입니다.

```
Num# (page 1222)(text[ , format[, dec_sep[ , thou_sep]]])
```

Text

Text()는 표현식을 숫자로 해석 가능한 경우에도 텍스트로 취급하도록 지정합니다.

```
Text(expr)
```

Time#

Time#()은 데이터 로드 스크립트 또는 운영 체제(서식 문자열을 지정하지 않은 경우)에 설정된 시간 서식을 사용하여 표현식을 시간 값으로 평가합니다..

```
Time# (page 1223)(text[, format])
```

Timestamp#

Timestamp#()는 데이터 로드 스크립트 또는 운영 체제(서식 문자열을 지정하지 않은 경우)에 설정된 타임스탬프 서식을 사용하여 표현식을 날짜 및 시간 값으로 평가합니다.

```
Timestamp# (page 1224)(text[, format])
```

관련 항목:

서식 지정 함수 (page 1184)

Date#

Date#는 두 번째 인수(지정된 경우)로 지정한 서식을 사용하여 표현식을 날짜로 평가합니다.

구문:

```
Date#(text[, format])
```

반환 데이터 유형: dual

인수:

인수

인수	설명
text	평가할 텍스트 문자열입니다.
format	평가할 텍스트 문자열의 서식을 설명하는 문자열입니다. 생략하면 데이터 로드 스크립트의 시스템 변수 또는 운영 체제에 설정된 날짜 서식이 사용됩니다.

예 및 결과:
다음 예에서는 **M/D/YYYY** 날짜 서식을 사용합니다. 날짜 서식은 데이터 로드 스크립트 맨 위에서 **SET DateFormat** 문으로 지정됩니다.

이 스크립트 예를 앱에 추가하고 실행합니다.

```
Load *,

Num(Date#(StringDate)) as Date;

LOAD * INLINE [

StringDate

8/7/97

8/6/1997

]
```

StringDate 및 **Date**를 차원으로 사용하는 테이블을 만드는 경우 결과는 다음과 같습니다.

결과

StringDate	Date
8/7/97	35649
8/6/1997	35648

Interval#

Interval#()은 운영 체제에서 기본 설정된 서식 또는 두 번째 인수(지정된 경우)로 지정된 서식을 사용하여 텍스트 표현식을 시간 간격으로 평가합니다.

구문:

```
Interval#(text[, format])
```

반환 데이터 유형: dual

인수:

인수

인수	설명
text	평가할 텍스트 문자열입니다.
format	문자열을 숫자 간격으로 변환할 때 사용할 예상 입력 서식을 설명하는 문자열입니다. 생략하는 경우 운영 체제에서 설정한 간단한 날짜 서식, 시간 서식 및 소수점 구분 기호가 사용됩니다.

interval# 함수는 텍스트 시간 간격을 동등한 숫자로 변환합니다.

예 및 결과:
아래 예에서는 다음과 같은 운영 체제 설정이 사용되고 있다고 가정합니다.

- 간단한 날짜 서식: YY-MM-DD
- 시간 서식: M/D/YY
- 숫자 소수점 구분 기호: .

결과

예	결과
Interval#(A, 'D hh:mm') 여기서 A='1 09:00'	1.375

Money#

Money#()는 로드 스크립트 또는 운영 체제(서식 문자열을 지정하지 않은 경우)에 설정된 서식을 사용하여 텍스트 문자열을 화폐 값으로 변환합니다. 사용자 지정 소수점 구분 기호 및 천 단위 구분 기호는 옵션 파라메타입니다.

구문:

```
Money#(text[, format[, dec_sep [, thou_sep ] ] ])
```

반환 데이터 유형: dual

인수:

인수

인수	설명
text	평가할 텍스트 문자열입니다.
format	문자열을 숫자 간격으로 변환할 때 사용할 예상 입력 서식을 설명하는 문자열입니다. 생략하면 운영 체제에 설정된 화폐 서식이 사용됩니다.
dec_sep	소수점 구분 기호를 지정하는 문자열입니다. 생략하면 데이터 로드 스크립트에 설정된 MoneyDecimalSep 값이 사용됩니다.
thou_sep	천 단위 구분 기호를 지정하는 문자열입니다. 생략하면 데이터 로드 스크립트에 설정된 MoneyThousandSep 값이 사용됩니다.

money# 함수는 일반적으로 **num#** 함수와 똑같이 동작하지만 화폐 서식에 대한 스크립트 변수 또는 통화에 대한 시스템 설정에서 소수점 및 천 단위 구분 기호의 기본값을 가져옵니다.

예 및 결과:
아래 예에서는 다음 두 가지 운영 체제 설정이 사용되고 있다고 가정합니다.

- 화폐 서식 기본 설정 1: kr # ##0,00
- 화폐 서식 기본 설정 2: $ #,##0.00

```
Money#(A , '# ##0,00 kr' )
```
여기서, A=35 648,37 kr

결과

결과	설정 1	설정 2
문자열	35 648.37 kr	35 648.37 kr
숫자	35648.37	3564837

```
Money#( A, ' $#', '.', ',' )
```
여기서, A= $35,648.37

결과

결과	설정 1	설정 2
문자열	$35,648.37	$35,648.37
숫자	35648.37	35648.37

Num#

Num#()은 텍스트 문자열을 숫자 값으로 해석합니다. 즉, 두 번째 매개 변수에 지정된 형식을 사용하여 입력 문자열을 숫자로 변환합니다. 두 번째 매개 변수를 생략하면 데이터 로드 스크립트에 설정된 소수 및 천 단위 구분 기호를 사용합니다. 사용자 지정 소수점 구분 기호 및 천 단위 구분 기호는 옵션 파라메타입니다.

구문:

```
Num#(text[, format[, dec_sep [, thou_sep ] ] ])
```

반환 데이터 유형: dual

Num#() 함수는 문자열과 숫자 값이 모두 포함된 이중 값을 반환합니다. 이 함수는 입력 표현식의 텍스트 표현을 가져와 숫자를 만듭니다. 숫자의 형식은 변경되지 않습니다. 출력은 입력과 같은 방식으로 형식이 지정됩니다.

인수:

인수

인수	설명
text	평가할 텍스트 문자열입니다.
format	첫 번째 매개 변수에 사용되는 숫자 서식을 지정하는 문자열입니다. 생략하면 데이터 로드 스크립트에 설정된 소수 및 천 단위 구분 기호가 사용됩니다.
dec_sep	소수점 구분 기호를 지정하는 문자열입니다. 생략하면 데이터 로드 스크립트에 설정된 변수 DecimalSep 값이 사용됩니다.
thou_sep	천 단위 구분 기호를 지정하는 문자열입니다. 생략하면 데이터 로드 스크립트에 설정된 변수 ThousandSep 값이 사용됩니다.

예 및 결과:
다음 표는 A의 다양한 값에 대한 *Num#(A, '#', '.', ',')*의 결과를 보여 줍니다.

결과

A	문자열 표시	숫자 값(여기에 소수점이 표시됨)
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()는 표현식을 숫자로 해석 가능한 경우에도 텍스트로 취급하도록 지정합니다.

구문:

```
Text (expr)
```

반환 데이터 유형: dual

```
Text( A )
```

여기서 A=1234

결과

문자열	숫자
1234	-

```
Text( pi( ) )
```

결과

문자열	숫자
3.1415926535898	-

Time#

Time#()은 데이터 로드 스크립트 또는 운영 체제(서식 문자열을 지정하지 않은 경우)에 설정된 시간 서식을 사용하여 표현식을 시간 값으로 평가합니다..

구문:

```
time#(text[, format])
```

반환 데이터 유형: dual

인수:

인수

인수	설명
text	평가할 텍스트 문자열입니다.
format	평가할 텍스트 문자열의 서식을 설명하는 문자열입니다. 생략하는 경우 운영 체제에서 설정한 간단한 날짜 서식, 시간 서식 및 소수점 구분 기호가 사용됩니다.

- 시간 서식 기본 설정 1: hh:mm:ss
- 시간 서식 기본 설정 2: hh.mm.ss

```
time#( A )
```
여기서 A=09:00:00

결과

결과	설정 1	설정 2
문자열:	09:00:00	09:00:00
숫자:	0.375	-

- 시간 서식 기본 설정 1: hh:mm:ss
- 시간 서식 기본 설정 2: hh.mm.ss

```
time#( A, 'hh.mm' )
```
여기서 A=09.00

결과

결과	설정 1	설정 2
문자열:	09.00	09.00
숫자:	0.375	0.375

Timestamp#

Timestamp#()는 데이터 로드 스크립트 또는 운영 체제(서식 문자열을 지정하지 않은 경우)에 설정된 타임스탬프 서식을 사용하여 표현식을 날짜 및 시간 값으로 평가합니다.

구문:

```
timestamp#(text[, format])
```

반환 데이터 유형: dual

인수:

인수

인수	설명
text	평가할 텍스트 문자열입니다.
format	평가할 텍스트 문자열의 서식을 설명하는 문자열입니다. 생략하는 경우 운영 체제에서 설정한 간단한 날짜 서식, 시간 서식 및 소수점 구분 기호가 사용됩니다. 타임스탬프에는 ISO 8601이 지원됩니다.

다음 예에서는 **M/D/YYYY** 날짜 서식을 사용합니다. 날짜 서식은 데이터 로드 스크립트 맨 위에서 **SET DateFormat** 문으로 지정됩니다.

이 스크립트 예를 앱에 추가하고 실행합니다.

```
Load *,
Timestamp(Timestamp#(String)) as TS;
LOAD * INLINE [
String
2015-09-15T12:13:14
1952-10-16T13:14:00+0200
1109-03-01T14:15
];
```

String 및 **TS**를 차원으로 사용하는 테이블을 생성하는 경우 결과는 다음과 같습니다.

결과

문자열	**TS**
2015-09-15T12:13:14	9/15/2015 12:13:14 PM
1952-10-16T13:14:00+0200	10/16/1952 11:14:00 AM
1109-03-01T14:15	3/1/1109 2:15:00 PM

8.16 인터 레코드 함수

인터 레코드 함수는 다음과 같이 사용됩니다.

- 데이터 로드 스크립트에서 현재 레코드를 평가하기 위해 이전에 로드된 데이터 레코드의 값이 필요할 때 사용됩니다.
- 차트 표현식에서 시각화의 데이터 셋에 있는 다른 값이 필요할 때 사용됩니다.

인터 레코드 차트 함수가 차트의 표현식에서 사용되는 경우 차트의 y 값에 대한 정렬 또는 테이블의 표현식 열에 의한 정렬은 허용되지 않습니다. 따라서 해당 정렬 옵션이 자동으로 비활성화됩니다. 시각화 또는 테이블에서 인터 레코드 차트 함수를 사용하면 시각화 정렬이 인터 레코드 함수에 대해 정렬된 입력으로 되돌아갑니다. 이 제한은 동등한 스크립트 함수가 있는 경우에는 적용되지 않습니다.

자체 참조 표현식 정의는 100행 미만의 테이블에서만 안정적으로 작성될 수 있지만 이는 Qlik 엔진이 실행 중인 하드웨어에 따라 달라질 수 있습니다.

행 함수

이 함수는 차트 표현식에서만 사용할 수 있습니다.

Above

Above()는 테이블의 열 세그먼트 내의 현재 행 위에 있는 행의 표현식을 평가합니다. 계산되는 행은 **offset**(있는 경우) 값에 따라 달라지며 기본적으로는 바로 위에 있는 행입니다. 테이블 이외의 차트인 경우, **Above()**는 해당 차트의 일반표 해당 부분의 현재 행 위에 있는 행을 평가합니다.

```
Above - 차트 함수([TOTAL [<fld{,fld}>]] expr [ , offset [,count]])
```

Below

Below()는 테이블의 열 세그먼트 내의 현재 행 아래에 있는 행의 표현식을 평가합니다. 계산되는 행은 **offset**(있는 경우) 값에 따라 달라지며 기본적으로는 바로 아래에 있는 행입니다. 테이블 이외의 차트인 경우, **Below()**는 해당 차트의 일반표 해당 부분의 현재 열 아래에 있는 행을 평가합니다.

```
Below - 차트 함수([TOTAL[<fld{,fld}>]] expression [ , offset [,count ]])
```

Bottom

Bottom()은 테이블의 열 세그먼트 내에 있는 마지막(맨 아래) 행의 표현식을 평가합니다. 계산되는 행은 **offset**(있는 경우) 값에 따라 달라지며 기본적으로는 맨 아래에 있는 행입니다. 테이블 이외의 차트에서는 해당 차트의 일반표 해당 부분에 있는 현재 열의 마지막 행을 평가합니다.

```
Bottom - 차트 함수([TOTAL[<fld{,fld}>]] expr [ , offset [,count ]])
```

Top

Top()은 테이블의 열 세그먼트 내에 있는 첫 번째(맨 위) 행의 표현식을 평가합니다. 계산되는 행은 **offset**(있는 경우) 값에 따라 달라지며 기본적으로는 맨 위에 있는 행입니다. 테이블 이외의 차트인 경우 **Top()**은 해당 차트의 일반표 동등 부분에 있는 현재 열의 첫 번째 행을 평가합니다.

```
Top - 차트 함수([TOTAL [<fld{,fld}>]] expr [ , offset [,count ]])
```

NoOfRows

NoOfRows()는 테이블 내 현재 열 세그먼트에 있는 행의 수를 반환합니다. 비트맵 차트의 경우, **NoOfRows()**는 해당 차트의 일반표 해당 부분에 있는 행 수를 반환합니다.

```
NoOfRows - 차트 함수([TOTAL])
```

열 함수

이 함수는 차트 표현식에서만 사용할 수 있습니다.

Column

Column()은 차원에 상관 없이 일반표의 **ColumnNo**에 해당하는 열에서 찾은 값을 반환합니다. 예를 들어 **Column(2)**는 두 번째 측정값 열의 값을 반환합니다.

```
Column - 차트 함수(ColumnNo)
```

Dimensionality

Dimensionality()는 현재 행의 차원 수를 반환합니다. 피벗 테이블의 경우 이 함수는 비집계 내용이 있는, 즉 부분합 또는 축소된 집계가 포함되지 않은 차원 열의 수를 반환합니다.

```
Dimensionality - 차트 함수 ( )
```

Secondarydimensionality

SecondaryDimensionality()는 비집계 내용이 있는, 즉 부분합 또는 축소된 집계가 포함되지 않은 차원 피벗 테이블 행의 수를 반환합니다. 이 함수는 가로 피벗 테이블 차원에 해당하는 **dimensionality()** 함수와 동등합니다.

```
SecondaryDimensionality - 차트 함수 ( )
```

필드 함수

FieldIndex

FieldIndex()는 **field_name** 필드의 필드 값 **value**의 위치를 반환합니다(로드 순서에 따라).

```
FieldIndex (field_name , value)
```

FieldValue

FieldValue()는 **field_name** 필드의 **elem_no** 위치에서 발견된 값을 반환합니다(로드 순서에 따라).

```
FieldValue (field_name , elem_no)
```

FieldValueCount

FieldValueCount()는 필드 내 고유 값의 수를 반환하는 **integer** 함수입니다.

```
FieldValueCount (field_name)
```

피벗 테이블 함수

이 함수는 차트 표현식에서만 사용할 수 있습니다.

After

After()는 피벗 테이블의 차원 값이 피벗 테이블의 행 세그먼트 내 현재 열 뒤의 열에 표시되므로 피벗 테이블의 차원 값으로 평가된 표현식의 값을 반환합니다.

```
After - 차트 함수([TOTAL] expression [ , offset [,n]])
```

Before

Before()는 피벗 테이블의 차원 값이 피벗 테이블의 행 세그먼트 내 현재 열 앞의 열에 표시되므로 피벗 테이블의 차원 값으로 평가된 표현식의 값을 반환합니다.

```
Before - 차트 함수([TOTAL] expression [ , offset [,n]])
```

First

First()는 피벗 테이블의 현재 행 세그먼트의 첫 번째 열에 표시되는 피벗 테이블의 차원 값으로 평가된 표현식의 값을 반환합니다. 이 함수는 피벗 테이블을 제외한 모든 차트 유형에서 NULL을 반환합니다.

```
First - 차트 함수([TOTAL] expression [ , offset [,n]])
```

Last

Last()는 피벗 테이블의 현재 행 세그먼트의 마지막 열에 표시되는 피벗 테이블의 차원 값으로 평가된 표현식의 값을 반환합니다. 이 함수는 피벗 테이블을 제외한 모든 차트 유형에서 NULL을 반환합니다.

```
Last - 차트 함수([TOTAL] expression [ , offset [,n]])
```

ColumnNo

ColumnNo()는 피벗 테이블의 현재 행 세그먼트 내에 있는 현재 열의 번호를 반환합니다. 첫 번째 열은 1번입니다.

```
ColumnNo - 차트 함수([TOTAL])
```

NoOfColumns

NoOfColumns()는 피벗 테이블 내 현재 행 세그먼트에 있는 열의 수를 반환합니다.

```
NoOfColumns - 차트 함수([TOTAL])
```

데이터 로드 스크립트 내의 인터 레코드 함수

Exists

Exists()는 특정 필드 값이 데이터 로드 스크립트의 필드에 이미 로드되었는지 여부를 판단합니다. 이 함수는 TRUE 또는 FALSE를 반환하므로 **LOAD** 문의 **where** 절 또는 **IF** 문에서 사용할 수 있습니다.

```
Exists (field_name [, expr])
```

LookUp

Lookup()은 이미 로드된 테이블을 조회하여 **match_field_name** 필드의 **match_field_value** 값의 첫 번째 발생 항목에 해당하는 **field_name**의 값을 반환합니다. 테이블은 현재 테이블이거나 이전에 로드한 다른 테이블일 수 있습니다.

```
LookUp (field_name, match_field_name, match_field_value [, table_name])
```

Peek

Peek()는 이미 로드된 행에 대한 테이블의 필드 값을 반환합니다. 테이블처럼 행 번호를 지정할 수 있습니다. 행 번호를 지정하지 않으면 이전에 마지막으로 로드된 레코드가 사용됩니다.

```
Peek (field_name[, row_no[, table_name ] ])
```

Previous

Previous()는 **where** 절로 인해 무시되지 않은 이전 입력 레코드의 데이터를 사용하여 **expr** 표현식의 값을 찾습니다. 이 함수는 내부 테이블의 첫 번째 레코드에 대해 NULL을 반환합니다.

```
Previous (page 1262)(expr)
```

관련 항목:

- *범위 함수 (page 1282)*

Above - 차트 함수

Above()는 테이블의 열 세그먼트 내의 현재 행 위에 있는 행의 표현식을 평가합니다. 계산되는 행은 **offset** (있는 경우) 값에 따라 달라지며 기본적으로는 바로 위에 있는 행입니다. 테이블 이외의 차트인 경우, **Above()**는 해당 차트의 일반표 해당 부분의 현재 행 위에 있는 행을 평가합니다.

구문:

```
Above([TOTAL] expr [ , offset [,count]])
```

반환 데이터 유형: dual

인수:

인수

인수	설명
expr	측정할 데이터가 포함된 표현식 또는 필드입니다.
offset	offsetn을 0보다 크게 지정하면 표현식의 평가가 현재 행보다 n 행 위로 이동됩니다. 오프셋을 0으로 지정하면 현재 행의 표현식이 평가됩니다. 오프셋을 음수로 지정하면 Above 함수가 상응하는 양수 오프셋이 있는 Below 함수와 동일하게 작동합니다.
count	세 번째 인수 **count**를 1보다 크게 지정하면 함수에서 **count** 값의 범위를 반환하며, 원래 셀에서 위쪽으로 계산하여 **count** 테이블 행 각각에 대해 하나의 값이 반환됩니다. 이러한 형식으로 이 함수를 특별 범위 함수의 인수로 활용할 수 있습니다. *범위 함수 (page 1282)*
TOTAL	테이블이 1차원이거나 **TOTAL** 한정자를 인수로 사용하는 경우 현재 열 세그먼트는 항상 전체 열과 동등합니다.

열 세그먼트의 첫 번째 행은 위에 다른 행이 없으므로 NULL 값이 반환됩니다.

열 세그먼트는 현재 정렬 순서에서 차원에 대해 동일한 값을 갖는 셀의 연속된 하위 집합으로 정의됩니다. 인터 레코드 차트 함수는 동등한 일반표 차트의 가장 오른쪽 차원을 제외한 열 세그먼트에서 계산됩니다. 차트에 차원이 하나만 있는 경우 또는 TOTAL 한정자가 지정된 경우는 표현식이 전체 테이블을 평가합니다.

테이블 또는 테이블과 동등한 것에 여러 세로 차원이 있을 경우 현재 열 세그먼트에는 필드 간 정렬 순서에서 마지막 차원이 표시되는 열을 제외하고 모든 차원 열 내의 현재 행과 같은 값을 가진 행만 포함됩니다.

제한 사항:

- 재귀 호출은 NULL을 반환합니다.
- 이 차트 함수가 차트의 표현식에서 사용되는 경우 차트의 y 값에 대한 정렬 또는 테이블의 표현식 열에 의한 정렬은 허용되지 않습니다. 따라서 해당 정렬 옵션이 자동으로 비활성화됩니다. 시각화 또는 테이블에서 이 차트 함수를 사용하면 시각화의 정렬이 이 함수에 대해 정렬된 입력으로 되돌아갑니다.

예 및 결과:

Example 1:

예 1의 테이블 시각화

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

이 예에서 보여 주는 테이블의 스크린샷에서, 테이블 시각화는 차원 **Customer**과 측정값 `Sum(Sales)` 및 `Above(Sum(Sales))`로부터 만들어집니다.

열 Above(Sum(Sales))는 위에 다른 행이 없으므로 **Astrida**가 포함된 **Customer** 행에 대해 NULL을 반환합니다. **Betacab** 행의 결과는 **Astrida**에 대한 Sum(Sales)의 값을 보여주며, **Canutility**의 결과는 **Betacab**에 대한 **Sum(Sales)**의 값을 보여줍니다.

레이블이 Sum(Sales)+Above(Sum(Sales))인 열의 경우, **Betacab**의 행에 행 **Betacab** + **Astrida**(539+587)의 **Sum(Sales)** 값의 덧셈 결과가 표시됩니다. **Canutility** 행의 결과는 **Canutility** + **Betacab**(683+539)의 **Sum (Sales)** 값의 덧셈 결과가 표시됩니다.

표현식 `Sum(Sales)+Above(Sum(Sales), 3)`를 사용하여 생성된 레이블이 Above offset 3인 계수에는 인수 **offset**이 3으로 설정되어 있으며, 현재 행의 세 번째 위 행에 있는 값을 취하는 결과를 갖습니다. 이는 현재 **Customer**의 **Sum(Sales)** 값을 세 행 위의 **Customer** 값에 추가합니다. 처음 세 **Customer** 행에서 반환되는 값은 Null입니다.

아래 테이블은 더 복잡한 계수를 보여줍니다. 하나는 `Sum(Sales)+Above(Sum(Sales))`에서 생성된 것이며, 하나는 레이블이 **Higher?**이고 `IF(Sum(Sales)>Above(Sum(Sales)), 'Higher')`에서 생성된 것입니다.

이 기능은 테이블 이외의 차트(예: 막대형 차트)에서 사용할 수 있습니다.

다른 차트 유형인 경우, 함수가 관련된 행을 쉽게 해석할 수 있도록 차트를 일반표 동등 부분으로 변환합니다.

Example 2:

이 예에 표시된 테이블의 스크린샷에는 시각화에 더 많은 차원이 추가되었습니다. **Month** 및 **Product**를 참조하십시오. 차원이 두 개 이상 있는 차트의 경우 **Above**, **Below**, **Top** 및 **Bottom** 함수를 포함하는 표현식의 결과는 Qlik Sense에서 열 차원이 정렬되는 순서에 따라 달라집니다. Qlik Sense에서는 차원의 마지막 정렬 결과인 열 세그먼트에 기반하여 해당 함수를 평가합니다. 열 정렬 순서는 속성 패널의 **정렬**에서 제어되며 이 순서가 반드시 테이블에 열이 표시되는 순서는 아닙니다.

아래 예 2의 테이블 시각화의 스크린샷에서 최종 정렬 차원은 **Month**이므로, **Above** 함수는 개월을 기준으로 평가합니다. 각 달(**Jan** ~ **Aug**) (열 세그먼트)에 해당하는 각 **Product** 값에 대해 일련의 결과가 존재합니다. 이는 일련의 다음 열 세그먼트 즉, 다음 **Product**에 대한 각 **Month**로 이어집니다. 각 **Product**에 대한 각 **Customer** 값에 해당하는 열 세그먼트 값이 존재하게 됩니다.

예 2의 테이블 시각화

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:

아래 예 3의 테이블 시각화의 스크린샷에서 최종 정렬 차원은 **Product**입니다. 이는 차원 Product을 속성 패널에 있는 정렬 탭의 3번 위치로 이동하여 수행됩니다. 각 **Product**에 대해 **Above** 함수가 평가되며, 제품이 **AA** 및 **BB** 두 가지뿐이므로 각 시리즈에서 Null이 아닌 결과는 하나뿐입니다. **Jan** 달에 대한 **BB** 행에서 **Above(Sum(Sales))**의 값은 46입니다. **AA**행의 값은 Null입니다. **AA** 위의 **Product** 값이 없으므로 모든 달의 각 AA 행의 값은 항상 Null이 됩니다. 두 번째 시리즈는 **Feb** 달의 **Customer** 값, **Astrida**의 **AA** 및 **BB**에 대해 평가됩니다. **Astrida**에 대해 모든 달이 평가되면, 두 번째 **Customer**Betacab 그리고 그 다음으로 시퀀스가 반복됩니다.

예 3의 테이블 시각화

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

예 4

Example 4:	**결과**
Above 함수는 범위 함수에 대한 입력으로 사용할 수 있습니다. 합니다(예: `RangeAvg (Above(Sum(Sales),1,3))`).	Above() 함수에 대한 인수에서 offset가 1로 설정되고 count이 3으로 설정됩니다. 이 함수는 열 세그먼트(행이 있는 곳)의 현재 행 바로 위의 세 행에서 표현식 Sum (Sales)의 결과를 찾습니다. 이 세 값은 제공된 숫자 범위에서 값의 평균을 구하는 RangeAvg() 함수에 대한 입력으로 사용됩니다. 차원으로 Customer가 포함된 테이블은 RangeAvg() 표현식에 대해 다음과 같은 결과를 제공합니다. Astrida - Betacab 587 Canutility 563 Divadip: 603

예에서 사용된 데이터:

```
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 '|');
```

관련 항목:

- *Below - 차트 함수 (page 1233)*

- *Bottom - 차트 함수 (page 1236)*
- *Top - 차트 함수 (page 1264)*
- *RangeAvg (page 1284)*

Below - 차트 함수

Below()는 테이블의 열 세그먼트 내의 현재 행 아래에 있는 행의 표현식을 평가합니다. 계산되는 행은 **offset**(있는 경우) 값에 따라 달라지며 기본적으로는 바로 아래에 있는 행입니다. 테이블 이외의 차트인 경우, **Below()**는 해당 차트의 일반표 해당 부분의 현재 열 아래에 있는 행을 평가합니다.

구문:

```
Below([TOTAL] expr [ , offset [,count ]])
```

반환 데이터 유형: dual

인수:

인수

인수	설명
expr	측정할 데이터가 포함된 표현식 또는 필드입니다.
offset	**offset**n을 1보다 크게 지정하면 표현식의 평가가 현재 행에서 n 행 아래로 이동됩니다. 오프셋을 0으로 지정하면 현재 행의 표현식이 평가됩니다. 오프셋을 음수로 지정하면 **Below** 함수가 상응하는 양수 오프셋이 있는 **Above** 함수와 동일하게 작동합니다.
count	세 번째 파라메타 **count**를 1보다 크게 지정하면 함수에서 **count** 값의 범위를 반환하며, 원래 셀에서 아래쪽으로 계산하여 **count** 테이블 행 각각에 대해 하나의 값이 반환됩니다. 이러한 형식으로 이 함수를 특별 범위 함수의 인수로 활용할 수 있습니다. *범위 함수 (page 1282)*
TOTAL	테이블이 1차원이거나 **TOTAL** 한정자를 인수로 사용하는 경우 현재 열 세그먼트는 항상 전체 열과 동등합니다.

열 세그먼트의 마지막 행 아래에 다른 행이 없으므로 NULL 값이 반환됩니다.

열 세그먼트는 현재 정렬 순서에서 차원에 대해 동일한 값을 갖는 셀의 연속된 하위 집합으로 정의됩니다. 인터 레코드 차트 함수는 동등한 일반표 차트의 가장 오른쪽 차원을 제외한 열 세그먼트에서 계산됩니다. 차트에 차원이 하나만 있는 경우 또는 TOTAL 한정자가 지정된 경우는 표현식이 전체 테이블을 평가합니다.

테이블 또는 테이블과 동등한 것에 여러 세로 차원이 있을 경우 현재 열 세그먼트에는 필드 간 정렬 순서에서 마지막 차원이 표시되는 열을 제외하고 모든 차원 열 내의 현재 행과 같은 값을 가진 행만 포함됩니다.

제한 사항:

- 재귀 호출은 NULL을 반환합니다.
- 이 차트 함수가 차트의 표현식에서 사용되는 경우 차트의 y 값에 대한 정렬 또는 테이블의 표현식 열에 의한 정렬은 허용되지 않습니다. 따라서 해당 정렬 옵션이 자동으로 비활성화됩니다. 시각화 또는 테이블에서 이 차트 함수를 사용하면 시각화의 정렬이 이 함수에 대해 정렬된 입력으로 되돌아갑니다.

예 및 결과:

Example 1:

예 1의 테이블 시각화

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	-	-	-	-

예 1의 스크린샷에 표시된 테이블에서, 테이블 시각화는 차원 **Customer**와 측정값 `Sum(Sales)` 및 `Below(Sum(Sales))`로부터 만들어집니다.

열 **Below(Sum(Sales))**는 아래에 다른 행이 없으므로 **Divadip**가 포함된 **Customer** 행에 대해 NULL을 반환합니다. **Canutility** 행의 결과는 **Divadip**에 대한 Sum(Sales)의 값을 보여주며, **Betacab**의 결과는 **Canutility**에 대한 **Sum(Sales)**의 값을 보여줍니다.

이 테이블은 더 복잡한 측정값을 보여주며, 레이블이 `Sum(Sales)+Below(Sum(Sales))`, **Below +Offset 3** 및 **Higher?**인 열에서 확인할 수 있습니다. 이 표현식은 아래 단락에 설명된 것처럼 작동합니다.

레이블이 **Sum(Sales)+Below(Sum(Sales))**인 열의 경우, **Astrida**의 행에 행 **Betacab** + **Astrida**(539+587)의 **Sum(Sales)** 값의 덧셈 결과가 표시됩니다. **Betacab** 행의 결과는 **Canutility** + **Betacab**(539+683)의 **Sum (Sales)** 값의 덧셈 결과가 표시됩니다.

표현식 `Sum(Sales)+Below(Sum(Sales), 3)`를 사용하여 만들어진 레이블이 **Below +Offset 3**인 측정값에는 인수 **offset**이 3으로 설정되어 있으며, 현재 행의 세 번째 아래 행에 있는 값을 취하는 결과를 갖습니다. 이는 현재 **Customer**의 **Sum(Sales)** 값을 세 행 아래의 **Customer** 값에 추가합니다. 맨 아래 세 **Customer** 행의 값은 Null입니다.

레이블이 **Higher?**인 계수는 표현식 `IF(Sum(Sales)>Below(Sum(Sales)), 'Higher')`에서 생성됩니다 . 이는 계수 **Sum(Sales)**의 현재 행 값과 그 아래 행의 값을 비교합니다. 현재 행의 값이 더 크다면 텍스트 "Higher"가 출력됩니다.

이 기능은 테이블 이외의 차트(예: 막대형 차트)에서 사용할 수 있습니다.

다른 차트 유형인 경우, 함수가 관련된 행을 쉽게 해석할 수 있도록 차트를 일반표 동등 부분으로 변환합니다.

차원이 두 개 이상 있는 차트의 경우 **Above**, **Below**, **Top** 및 **Bottom** 함수를 포함하는 표현식의 결과는 Qlik Sense에서 열 차원이 정렬되는 순서에 따라 달라집니다. Qlik Sense에서는 차원의 마지막 정렬 결과인 열 세그먼트에 기반하여 해당 함수를 평가합니다. 열 정렬 순서는 속성 패널의 **정렬**에서 제어되며 이 순서가 반드시 테이블에 열이 표시되는 순서는 아닙니다.자세한 내용은 **Above** 함수의 예: 2를 참조하십시오.

예 2

Example 2:	**결과**
Below 함수는 범위 함수에 대한 입력으로 사용할 수 있습니다. 합니다(예: `RangeAvg (Below(Sum (Sales),1,3))`).	**Below()** 함수에 대한 인수에서 offset가 1로 설정되고 count이 3으로 설정됩니다. 이 함수는 열 세그먼트(행이 있는 곳)의 현재 행 바로 아래의 세 행에서 표현식 **Sum(Sales)**의 결과를 찾습니다. 이 세 값은 제공된 숫자 범위에서 값의 평균을 구하는 RangeAvg() 함수에 대한 입력으로 사용됩니다. 차원으로 **Customer**가 포함된 테이블은 RangeAvg() 표현식에 대해 다음과 같은 결과를 제공합니다. Astrida 659.67 Betacab 720 Canutility 757 Divadip: -

예에서 사용된 데이터:

```
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 '|');
```

관련 항목:

- *Above - 차트 함수 (page 1228)*
- *Bottom - 차트 함수 (page 1236)*
- *Top - 차트 함수 (page 1264)*
- *RangeAvg (page 1284)*

Bottom - 차트 함수

Bottom()은 테이블의 열 세그먼트 내에 있는 마지막(맨 아래) 행의 표현식을 평가합니다. 계산되는 행은 **offset**(있는 경우) 값에 따라 달라지며 기본적으로는 맨 아래에 있는 행입니다. 테이블 이외의 차트에서는 해당 차트의 일반표 해당 부분에 있는 현재 열의 마지막 행을 평가합니다.

구문:

```
Bottom([TOTAL] expr [ , offset [,count ]])
```

반환 데이터 유형: dual

인수:

인수

인수	설명
expr	측정할 데이터가 포함된 표현식 또는 필드입니다.
offset	**offset**n을 1보다 크게 지정하면 표현식의 평가가 맨 아래 행에서 n 행 위로 이동됩니다. 오프셋을 음수로 지정하면 **Bottom** 함수가 상응하는 양수 오프셋이 있는 **Top** 함수와 동일하게 작동합니다.
count	세 번째 파라메타 **count**를 1보다 크게 지정하면 이 함수에서 하나가 아닌 **count**의 범위 값을 반환하며, 현재 열 세그먼트의 마지막 **count** 행 각각에 대해 하나의 값을 반환합니다. 이러한 형식으로 이 함수를 특별 범위 함수의 인수로 활용할 수 있습니다. *범위 함수 (page 1282)*
TOTAL	테이블이 1차원이거나 **TOTAL** 한정자를 인수로 사용하는 경우 현재 열 세그먼트는 항상 전체 열과 동등합니다.

열 세그먼트는 현재 정렬 순서에서 차원에 대해 동일한 값을 갖는 셀의 연속된 하위 집합으로 정의됩니다. 인터 레코드 차트 함수는 동등한 일반표 차트의 가장 오른쪽 차원을 제외한 열 세그먼트에서 계산됩니다. 차트에 차원이 하나만 있는 경우 또는 TOTAL 한정자가 지정된 경우는 표현식이 전체 테이블을 평가합니다.

테이블 또는 테이블과 동등한 것에 여러 세로 차원이 있을 경우 현재 열 세그먼트에는 필드 간 정렬 순서에서 마지막 차원이 표시되는 열을 제외하고 모든 차원 열 내의 현재 행과 같은 값을 가진 행만 포함됩니다.

제한 사항:

- 재귀 호출은 NULL을 반환합니다.
- 이 차트 함수가 차트의 표현식에서 사용되는 경우 차트의 y 값에 대한 정렬 또는 테이블의 표현식 열에 의한 정렬은 허용되지 않습니다. 따라서 해당 정렬 옵션이 자동으로 비활성화됩니다. 시각화 또는 테이블에서 이 차트 함수를 사용하면 시각화의 정렬이 이 함수에 대해 정렬된 입력으로 되돌아갑니다.

예 및 결과:

예 1의 테이블 시각화

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

이 예에서 보여 주는 테이블의 스크린샷에서, 테이블 시각화는 차원 **Customer**과 측정값 `Sum(Sales)` 및 `Bottom(Sum(Sales))`로부터 만들어집니다.

Bottom(Sum(Sales)) 열은 모든 행에 대해 맨 아래 행 **Divadip**의 값인 757을 반환합니다.

이 테이블은 더 복잡한 계수를 보여줍니다. 하나는 `Sum(Sales)+Bottom(Sum(Sales))`에서 생성된 것이며, 하나는 레이블이 **Bottom offset 3**이고 표현식 `Sum(Sales)+Bottom(Sum(Sales), 3)`에서 생성된 것으로 인수 **offset**이 3으로 설정되었습니다. 이는 현재 행의 **Sum(Sales)** 값을 맨 아래 행에서 세 번째 행의 값에 추가하는 것이며, 현재 행에 **Betacab**의 값을 더하는 것과 같습니다.

2

이 예에 표시된 테이블의 스크린샷에는 시각화에 더 많은 차원이 추가되었습니다. **Month** 및 **Product**의 측정값 2개를 사용합니다. 차원이 두 개 이상 있는 차트의 경우 **Above**, **Below**, **Top** 및 **Bottom** 함수를 포함하는 표현식의 결과는 Qlik Sense에서 열 차원이 정렬되는 순서에 따라 달라집니다. Qlik Sense에서는 차원의 마지막 정렬 결과인 열 세그먼트에 기반하여 해당 함수를 평가합니다. 열 정렬 순서는 속성 패널의 **정렬**에서 제어되며 이 순서가 반드시 테이블에 열이 표시되는 순서는 아닙니다.

첫 번째 테이블에서 표현식은 **Month** 기준으로 평가되며, 두 번째 테이블에서 표현식은 **Product** 기준으로 평가됩니다. 계수 **End value**에는 표현식 `Bottom(Sum(Sales))`이 포함됩니다. **Month**의 맨 아래 행은 Dec이며, 스크린샷에 표시된 **Product** 값 모두의 Dec에 대한 값은 22입니다. (스크린샷에는 공간 절약을 위해 일부 행이 편집되었습니다.)

예 2의 첫 번째 테이블입니다. Month(Dec)에 기반한 End value 측정값의 Bottom 값입니다.

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

예 2의 두 번째 테이블입니다. Product(Astrida의 BB)에 기반한 End value 측정값의 Bottom 값입니다.

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

자세한 내용은 **Above** 함수의 예: 2를 참조하십시오.

예 3

3	결과
Bottom 함수는 범위 함수에 대한 입력으로 사용할 수 있습니다. 합니다(예: `RangeAvg (Bottom(Sum (Sales),1,3))`).	**Bottom()** 함수에 대한 인수에서 offset는 1로 설정되고 count은 3으로 설정됩니다. 이 함수는 열 세그먼트에 있는 맨 아래 행 위의 행에서 시작하는 세 행(offset=1이므로)과 그 위의 두 행(행이 있는 곳)에 대한 표현식 **Sum(Sales)**의 결과를 찾습니다. 이 세 값은 제공된 숫자 범위에서 값의 평균을 구하는 RangeAvg() 함수에 대한 입력으로 사용됩니다. 차원으로 **Customer**가 포함된 테이블은 RangeAvg() 표현식에 대해 다음과 같은 결과를 제공합니다. 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 '|');
```

관련 항목:

Top - 차트 함수 (page 1264)

Column - 차트 함수

Column()은 차원에 상관 없이 일반표의 **ColumnNo**에 해당하는 열에서 찾은 값을 반환합니다. 예를 들어 **Column(2)**는 두 번째 측정값 열의 값을 반환합니다.

구문:

```
Column(ColumnNo)
```

반환 데이터 유형: dual

인수:

인수

인수	설명
ColumnNo	측정값이 포함된 테이블에 있는 열의 열 번호. *Column() 함수는 차원 열을 무시합니다.*

제한 사항:

- 재귀 호출은 NULL을 반환합니다.
- **ColumnNo**가 계수가 없는 열을 참조하는 경우, NULL 값이 반환됩니다.
- 이 차트 함수가 차트의 표현식에서 사용되는 경우 차트의 y 값에 대한 정렬 또는 테이블의 표현식 열에 의한 정렬은 허용되지 않습니다. 따라서 해당 정렬 옵션이 자동으로 비활성화됩니다. 시각화 또는 테이블에서 이 차트 함수를 사용하면 시각화의 정렬이 이 함수에 대해 정렬된 입력으로 되돌아갑니다.

예 및 결과:

총 판매 비율

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
B	BB	10	5	50	505	9.90
B	CC	20	2	40	505	7.92
B	DD	25	-	0	505	0.00

Customer	Product	UnitPrice	UnitSales	Order Value	Total Sales Value	% Sales
C	AA	15	8	120	505	23.76
C	CC	19	-	0	505	0.00

선택한 고객의 판매 비율

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

예 및 결과

예	결과
Order Value가 표현식 `Sum (UnitPrice*UnitSales)`과 함께 계수로 테이블에 추가됩니다. Total Sales Value가 다음 표현식과 함께 계수로 추가됩니다. `Sum(TOTAL UnitPrice*UnitSales)` % Sales가 표현식 `100*Column(1)/Column(2)`과 함께 계수로 추가됩니다.	Column(1)의 결과는 첫 번째 계수 열인 Order Value 열에서 가져옵니다. Column(2)의 결과는 두 번째 계수 열인 Total Sales Value에서 가져옵니다. 예제 *총 판매 비율 (page 1240)*의 % Sales 열에 있는 결과를 참조하십시오.
Customer A를 선택합니다.	선택 내용에 따라 Total Sales Value, 그리고 %Sales가 변경됩니다. 예제 *선택한 고객의 판매 비율 (page 1241)*를 확인하십시오.

예에서 사용된 데이터:

```
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 - 차트 함수

Dimensionality()는 현재 행의 차원 수를 반환합니다. 피벗 테이블의 경우 이 함수는 비집계 내용이 있는, 즉 부분합 또는 축소된 집계가 포함되지 않은 차원 열의 수를 반환합니다.

구문:

```
Dimensionality ( )
```

반환 데이터 유형: 정수

제한 사항:

이 함수는 차트 내에서만 사용 가능합니다. 피벗 테이블을 제외한 모든 차트 유형에서 이 함수는 0이 되는 합계를 제외한 모든 행의 차원 수를 반환합니다.

이 차트 함수가 차트의 표현식에서 사용되는 경우 차트의 y 값에 대한 정렬 또는 테이블의 표현식 열에 의한 정렬은 허용되지 않습니다. 따라서 해당 정렬 옵션이 자동으로 비활성화됩니다. 시각화 또는 테이블에서 이 차트 함수를 사용하면 시각화의 정렬이 이 함수에 대해 정렬된 입력으로 되돌아갑니다.

예: Dimensionality를 사용하는 차트 표현식

예: 차트 표현식

Dimensionality() 함수는 집계되지 않은 데이터가 있는 행의 차원 수에 따라 다른 셀 서식을 적용하려는 차트 표현식으로 피벗 테이블과 함께 사용할 수 있습니다. 이 예에서는 Dimensionality() 함수를 사용하여 주어진 조건과 일치하는 테이블 셀에 배경색을 적용합니다.

로드 스크립트

데이터 로드 편집기에서 다음 데이터를 인라인 로드로 로드하여 아래 차트 표현식 예를 만듭니다.

```
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 ',');
```

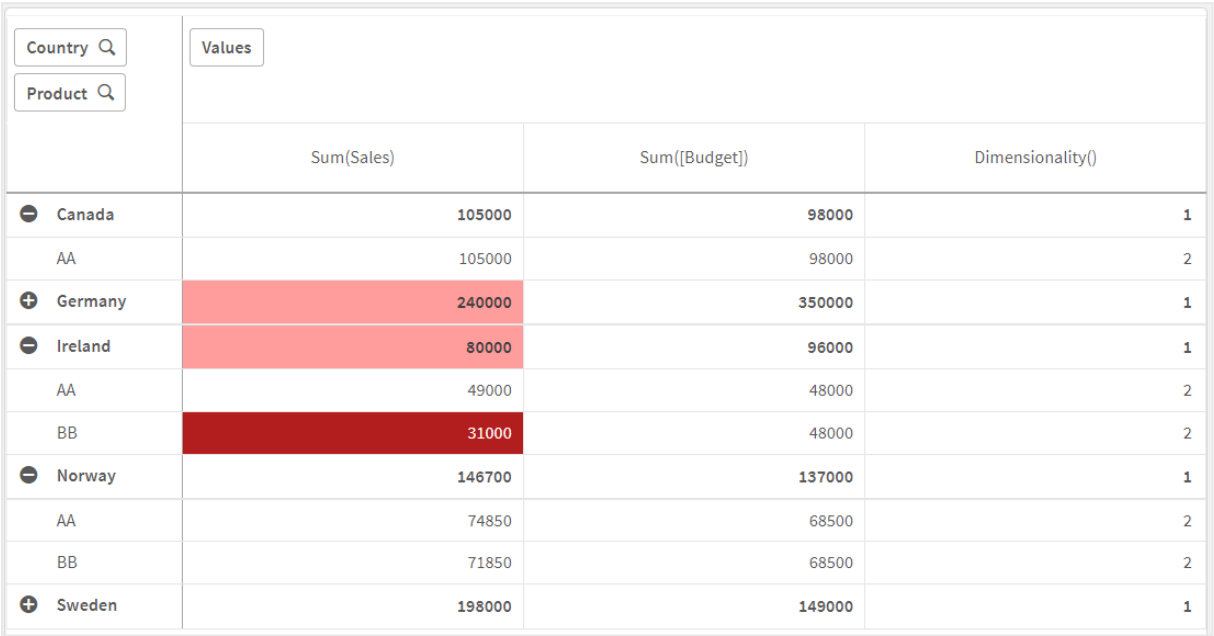
차트 표현식

Country 및 **Product**를 차원으로 사용하여 Qlik Sense 시트에서 피벗 테이블 시각화를 만듭니다. **Sum(Sales)**, **Sum(Budget)** 및 **Dimensionality()**를 측정값으로 추가합니다.

속성 패널에서 **Sum(Sales)** 측정값에 대한 **배경색 표현식**으로 다음 표현식을 입력합니다.

```
If(Dimensionality()=1 and Sum(Sales)<Sum(Budget),RGB(255,156,156),
If(Dimensionality()=2 and Sum(Sales)<Sum(Budget),RGB(178,29,29)
))
```

결과:



Country
Product
Values

	Sum(Sales)	Sum([Budget])	Dimensionality()
Canada	105000	98000	1
AA	105000	98000	2
Germany	240000	350000	1
Ireland	80000	96000	1
AA	49000	48000	2
BB	31000	48000	2
Norway	146700	137000	1
AA	74850	68500	2
BB	71850	68500	2
Sweden	198000	149000	1

설명

표현식 `If(Dimensionality()=1 and Sum(Sales)<Sum(Budget),RGB(255,156,156), If(Dimensionality ()=2 and Sum(Sales)<Sum(Budget),RGB(178,29,29)))`에는 각 제품에 대한 dimensionality 값과 Sum(Sales) 및 Sum(Budget)을 확인하는 조건문이 포함됩니다. 조건이 충족되면 Sum(Sales) 값에 배경색이 적용됩니다.

Exists

Exists()는 특정 필드 값이 데이터 로드 스크립트의 필드에 이미 로드되었는지 여부를 판단합니다. 이 함수는 TRUE 또는 FALSE를 반환하므로 **LOAD** 문의 **where** 절 또는 **IF** 문에서 사용할 수 있습니다.

> ***Not Exists()****를 사용하여 필드 값이 로드되었는지 확인할 수도 있지만 where 절에서* ***Not Exists()*** *를 사용할 때는 주의해야 합니다.* ***Exists()*** *함수는 이전에 로드한 테이블과 현재 테이블에서 이전에 로드한 값을 둘 다 테스트합니다. 따라서 처음 발견되는 값이 로드됩니다. 두 번째 값이 발견될 때 값은 이미 로드되어 있습니다. 자세한 내용은 예제를 참조하십시오.*

구문:

```
Exists(field_name [, expr])
```

반환 데이터 유형: 부울

인수:

인수

인수	설명
field_name	값을 검색할 필드의 이름입니다. 따옴표 없이 명시적 필드 이름을 사용할 수 있습니다. 스크립트에서 필드가 이미 로드되어 있어야 합니다. 즉, 스크립트에서 나중 절에 로드된 필드를 참조할 수 없습니다.
expr	존재하는지 확인하려는 값입니다. 현재 load 문에서 하나 또는 여러 개의 필드를 참조하는 명시적 값 또는 표현식을 사용할 수 있습니다. *현재 load 문에 포함되지 않은 필드는 참조할 수 없습니다.* 이 인수는 선택 사항입니다. 이 인수를 생략하면 함수는 현재 레코드에 **field_name** 값이 이미 존재하는지 확인합니다.

예 및 결과:

예 1

```
Exists (Employee)
```

현재 레코드의 **Employee** 필드 값이 이전에 읽은 레코드 중 해당 필드가 포함된 레코드에 이미 존재할 경우 -1(True)을 반환합니다.

`Exists (Employee, Employee)` 및 `Exists (Employee)` 문은 동등합니다.

예 2

```
Exists(Employee, 'Bill')
```

필드 값 **'Bill'**이 **Employee** 필드의 현재 내용에서 발견되는 경우 -1(True)을 반환합니다.

예 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;
```

이 결과는 Employee 및 Address 차원을 사용하여 테이블 시각화에서 사용할 수 있는 테이블에 표시됩니다.

where 절: `where Exists (Employee)`는 Citizens 테이블의 이름 중 Employees에도 있는 이름만 새 테이블로 로드합니다. Drop 문은 혼란을 피하기 위해 테이블 Employees를 제거합니다.

결과

Employee	Address
Bill	New York
John	Miami
Steve	Chicago

예 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;
```

where 절에 not이 포함됩니다(`where not Exists (Employee)`).

이는 Citizens 테이블의 이름 중 Employees에 없는 이름만 새 테이블로 로드됨을 의미합니다.

Citizens 테이블에 Lucy의 경우 두 개의 값이 있지만 결과 테이블에는 하나만 포함됩니다. Lucy 값이 있는 첫 번째 행을 로드하면 Employee 필드에 포함됩니다. 따라서 두 번째 줄을 확인하면 값이 이미 존재합니다.

결과

Employee	**주소**
Mary	London
Lucy	Madrid

예 5

이 예는 모든 값을 로드하는 방법을 보여 줍니다.

```
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;
```

Lucy에 대한 모든 값을 가져올 수 있도록 두 가지 사항이 변경되었습니다.

- Name으로 Employee 이름이 변경된 Employees 테이블에 대한 선행 LOAD가 삽입되었습니다.
  `Load Employee As Name;`
- Citizens에서 Where 조건이 다음으로 변경되었습니다.
  `not Exists (Name, Employee).`

그러면 Name 및 Employee에 대한 필드가 만들어집니다. Lucy가 있는 두 번째 행을 선택하면 Name에 여전히 존재하지 않습니다.

결과

Employee	**주소**
Mary	London

Employee	주소
Lucy	Madrid
Lucy	Paris

FieldIndex

FieldIndex()는 **field_name** 필드의 필드 값 **value**의 위치를 반환합니다(로드 순서에 따라).

구문:

```
FieldIndex(field_name , value)
```

반환 데이터 유형: 정수

인수:

인수

인수	설명
field_name	색인이 필요한 필드의 이름. 예를 들어, 테이블에 있는 열입니다, 문자열 값으로 지정해야 합니다. 따라서 필드 이름은 작은따옴표로 묶어야 합니다.
value	**field_name** 필드의 값.

제한 사항:

- **field_name** 필드의 필드 값 중에서 **value**를 찾을 수 없는 경우 0이 반환됩니다.
- 이 차트 함수가 차트의 표현식에서 사용되는 경우 차트의 y 값에 대한 정렬 또는 테이블의 표현식 열에 의한 정렬은 허용되지 않습니다. 따라서 해당 정렬 옵션이 자동으로 비활성화됩니다. 시각화 또는 테이블에서 이 차트 함수를 사용하면 시각화의 정렬이 이 함수에 대해 정렬된 입력으로 되돌아갑니다. 이 제한은 동등한 스크립트 함수에는 적용되지 않습니다.

예 및 결과:

다음 예는 **First name** 테이블의 **Names** 필드를 사용합니다.

예 및 결과

예	결과
데이터 예를 앱에 추가하고 실행합니다.	샘플 데이터에서와 마찬가지로 테이블 **Names**가 로드됩니다.
차트 함수: 차원 First name이 포함된 테이블에 측정값으로 다음을 추가합니다.	

예	결과
`FieldIndex ('First name','John')`	1이며, 'John'이 **First name** 필드의 로드 순서에서 처음으로 나오기 때문입니다. 필터 창은 로드 순서가 아니라 알파벳 순으로 정렬되므로 **John**이 맨 위에서 두 번째로 표시된다는 점에 유의하십시오.
`FieldIndex ('First name','Peter')`	4이며, **FieldIndex()**가 로드 순서에서 첫 번째로 발견되는 하나의 값만을 반환하기 때문입니다.
스크립트 함수: 예제 데이터에서와 마찬가지로 테이블 **Names**가 로드된 것으로 가정합니다.	

`John1:`
`Load FieldIndex('First name','John') as MyJohnPos`
`Resident Names;`
	`MyJohnPos=1`이며, 'John'이 **First name** 필드의 로드 순서에서 처음으로 나오기 때문입니다. 필터 창은 로드 순서가 아니라 알파벳 순으로 정렬되므로 **John**이 맨 위에서 두 번째로 표시된다는 점에 유의하십시오.

`Peter1:`
`Load FieldIndex('First name','Peter') as MyPeterPos`
`Resident Names;`
	`MyPeterPos=4`이며, **FieldIndex()**가 로드 순서에서 첫 번째로 발견되는 하나의 값만을 반환하기 때문입니다.

데이터 사용 예:

```
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()는 **field_name** 필드의 **elem_no** 위치에서 발견된 값을 반환합니다(로드 순서에 따라).

구문:

```
FieldValue(field_name , elem_no)
```

반환 데이터 유형: dual

인수:

인수

인수	설명
field_name	값이 필요한 필드의 이름. 예를 들어, 테이블에 있는 열입니다, 문자열 값으로 지정해야 합니다. 따라서 필드 이름은 작은따옴표로 묶어야 합니다.
elem_no	값이 반환된 로드 순서에 따른 필드의 위치(요소) 번호. 이는 테이블에 있는 행에 해당할 수 있지만, 요소(행)가 로드되는 순서에 따라 결정됩니다.

제한 사항:

- **elem_no**가 필드 값의 수보다 클 경우 NULL이 반환됩니다.
- 이 차트 함수가 차트의 표현식에서 사용되는 경우 차트의 y 값에 대한 정렬 또는 테이블의 표현식 열에 의한 정렬은 허용되지 않습니다. 따라서 해당 정렬 옵션이 자동으로 비활성화됩니다. 시각화 또는 테이블에서 이 차트 함수를 사용하면 시각화의 정렬이 이 함수에 대해 정렬된 입력으로 되돌아갑니다. 이 제한은 동등한 스크립트 함수에는 적용되지 않습니다.

예

로드 스크립트

데이터 로드 편집기에서 다음 데이터를 인라인 로드로 로드하여 아래 예를 만듭니다.

```
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;
```

시각화 만들기

Qlik Sense 시트에 테이블 시각화를 만듭니다. 테이블에 **First name**, **MyPos1** 및 **MyPos2** 필드를 추가합니다.

결과

First name	MyPos1	MyPos2
Jane	John	Jane
John	John	Jane
Mark	John	Jane
Peter	John	Jane
Sue	John	Jane

설명

First name 필드의 로드 순서로 John이 맨 처음에 나타나기 때문에 모든 이름에 대한 **FieldValue('First name','1')**의 **MyPos1** 값은 John이 됩니다. 필터 창은 로드 순서가 아니라 알파벳 순으로 정렬되므로 John이 맨 위에서 두 번째로 Jane 다음에 표시된다는 점에 유의하십시오.

First name 필드의 로드 순서로 Jane이 5번째에 나타나기 때문에 모든 이름에 대한 **FieldValue('First name','1')**의 **MyPos2** 값은 Jane이 됩니다.

FieldValueCount

FieldValueCount()는 필드 내 고유 값의 수를 반환하는 **integer** 함수입니다.

부분 로드는 데이터에서 값을 제거할 수 있으며 이는 반환되는 숫자에 반영되지 않습니다. 반환되는 숫자는 초기 다시 로드 또는 모든 후속 부분 로드에서 로드된 모든 고유 값에 해당합니다.

이 차트 함수가 차트의 표현식에서 사용되는 경우 차트의 y 값에 대한 정렬 또는 테이블의 표현식 열에 의한 정렬은 허용되지 않습니다. 따라서 해당 정렬 옵션이 자동으로 비활성화됩니다. 시각화 또는 테이블에서 이 차트 함수를 사용하면 시각화의 정렬이 이 함수에 대해 정렬된 입력으로 되돌아갑니다. 이 제한은 동등한 스크립트 함수에는 적용되지 않습니다.

구문:

```
FieldValueCount(field_name)
```

반환 데이터 유형: 정수

인수:

인수

인수	설명
field_name	값이 필요한 필드의 이름. 예를 들어, 테이블에 있는 열입니다, 문자열 값으로 지정해야 합니다. 따라서 필드 이름은 작은따옴표로 묶어야 합니다.

예 및 결과:

다음 예는 **Names** 테이블의 **First name** 필드를 사용합니다.

예 및 결과

예	결과
데이터 예를 앱에 추가하고 실행합니다.	샘플 데이터에서와 마찬가지로 테이블 **Names**가 로드됩니다.
차트 함수: 차원 First name이 포함된 테이블에 측정값으로 다음을 추가합니다.	
`FieldValueCount('First name')`	5이며 **Peter**가 두 번 나오기 때문입니다.
`FieldValueCount('Initials')`	6이며 **Initials**에 고유 값만이 있기 때문입니다.
스크립트 함수: 예제 데이터에서와 마찬가지로 테이블 **Names**가 로드된 것으로 가정합니다.	

`FieldCount1:`
`Load FieldValueCount('First name') as MyFieldCount1`
`Resident Names;`
	`MyFieldCount1=5`이며, 'Peter'이 두 번 나오기 때문입니다.

`FieldCount2:`
`Load FieldValueCount('Initials') as MyInitialsCount1`
`Resident Names;`
	`MyFieldCount1=6`이며, 'Initials'에 고유 값만이 있기 때문입니다.

예에서 사용된 데이터:

```
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()은 이미 로드된 테이블을 조회하여 **match_field_name** 필드의 **match_field_value** 값의 첫 번째 발생 항목에 해당하는 **field_name**의 값을 반환합니다. 테이블은 현재 테이블이거나 이전에 로드한 다른 테이블일 수 있습니다.

구문:

```
lookup(field_name, match_field_name, match_field_value [, table_name])
```

반환 데이터 유형: dual

인수:

인수

인수	설명
field_name	반환 값이 필요한 필드의 이름입니다. 입력 값은 문자열(예: 따옴표로 묶은 리터럴)로 지정해야 합니다.
match_field_name	**match_field_value**를 조회할 필드의 이름입니다. 입력 값은 문자열(예: 따옴표로 묶은 리터럴)로 지정해야 합니다.
match_field_value	**match_field_name** 필드에서 조회할 값입니다.
table_name	값을 조회할 테이블의 이름입니다. 입력 값은 문자열(예: 따옴표로 묶은 리터럴)로 지정해야 합니다. **table_name**을 생략하면 현재 테이블이 사용됩니다.

따옴표가 없는 인수는 현재 테이블을 참조합니다. 다른 테이블을 참조하려면 작은따옴표 안에 해당 인수를 넣으십시오.

제한 사항:

테이블이 조인과 같은 복잡한 연산의 결과일 경우 순서가 잘 정의되지 않으며, 그렇지 않을 경우 로드 순서가 검색 순서가 됩니다. **field_name** 및 **match_field_name**은 **table_name**으로 지정된 테이블과 동일한 테이블 내의 필드여야 합니다.

일치 항목이 발견되지 않으면 NULL이 반환됩니다.

예

로드 스크립트

데이터 로드 편집기에서 다음 데이터를 인라인 로드로 로드하여 아래 예를 만듭니다.

```
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;
```

시각화 만들기

Qlik Sense 시트에 테이블 시각화를 만듭니다. 테이블에 **ProductID**, **InvoiceID**, **CustomerID**, **Units** 및 **CategoryID** 필드를 추가합니다.

결과

결과 테이블

ProductID	InvoiceID	CustomerID	단위	CategoryID
1	1	Astrida	8	1
2	1	Astrida	6	1
3	2	Betacab	10	2
3	3	Divadip	5	2
4	4	Divadip	10	3

설명

샘플 데이터에서는 다음과 같은 형식으로 **Lookup()** 함수를 사용합니다.

```
Lookup('Category', 'ProductID', ProductID, 'ProductList')
```

ProductList 테이블이 먼저 로드됩니다.

Lookup() 함수는 **OrderData** 테이블을 만드는 데 사용됩니다. 세 번째 인수가 **ProductID**로 지정됩니다. 둘러싼 작은따옴표에 지정된 대로 **ProductList**의 두 번째 인수 **'ProductID'**에서 값을 조회할 필드입니다.

이 함수는 **CategoryID**로 로드된 '**Category**'(**ProductList** 테이블 내)의 값을 반환합니다.

drop 문은 불필요한 **ProductList** 테이블을 데이터 모델에서 삭제하며 결과 **OrderData** 테이블은 남겨둡니다.

Lookup() 함수는 유연하므로 이전에 로드한 테이블에 액세스할 수 있습니다. 하지만 Applymap() 함수와 비교하여 속도가 느립니다.

관련 항목:

ApplyMap (page 1275)

NoOfRows - 차트 함수

NoOfRows()는 테이블 내 현재 열 세그먼트에 있는 행의 수를 반환합니다. 비트맵 차트의 경우, **NoOfRows()**는 해당 차트의 일반표 해당 부분에 있는 행 수를 반환합니다.

테이블 또는 테이블과 동등한 것에 여러 세로 차원이 있을 경우 현재 열 세그먼트에는 필드 간 정렬 순서에서 마지막 차원이 표시되는 열을 제외하고 모든 차원 열 내의 현재 행과 같은 값을 가진 행만 포함됩니다.

이 차트 함수가 차트의 표현식에서 사용되는 경우 차트의 y 값에 대한 정렬 또는 테이블의 표현식 열에 의한 정렬은 허용되지 않습니다. 따라서 해당 정렬 옵션이 자동으로 비활성화됩니다. 시각화 또는 테이블에서 이 차트 함수를 사용하면 시각화의 정렬이 이 함수에 대해 정렬된 입력으로 되돌아갑니다.

구문:

```
NoOfRows([TOTAL])
```

반환 데이터 유형: 정수

인수:

인수

인수	설명
TOTAL	테이블이 1차원이거나 **TOTAL** 한정자를 인수로 사용하는 경우 현재 열 세그먼트는 항상 전체 열과 동등합니다.

예: NoOfRows를 사용한 차트 표현식

예 - 차트 표현식

로드 스크립트

데이터 로드 편집기에서 다음 데이터를 인라인 로드로 로드하여 아래 차트 표현식 예를 만듭니다.

```
        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 '|');
```

차트 표현식

Region 및 **SubRegion**를 차원으로 사용하여 Qlik Sense 시트에 테이블 시각화를 만듭니다. RowNo( ), NoOfRows() 및 NoOfRows(Total)를 측정값으로 추가합니다.

결과

Region	SubRegion	RowNo()	NoOfRows()	NoOfRows (Total)
Africa	Eastern	1	2	9
Africa	Western	2	2	9
Americas	Central	1	2	9
Americas	Northern	2	2	9
Asia	Eastern	1	1	9
Europe	Eastern	1	3	9
Europe	Northern	2	3	9
Eurrope	Western	3	3	9
Oceania	Australia	1	1	9

설명

이 예에서 정렬 순서는 첫 번째 차원인 Region을 기준으로 합니다. 결과적으로 각 열 세그먼트는 동일한 값을 갖는 지역 그룹으로 구성됩니다(예: 아프리카).

RowNo() 열은 각 열 세그먼트에 대한 행 번호를 보여 줍니다. 예를 들어 아프리카 지역에 대한 두 개의 행이 있습니다. 그러면 다음 열 세그먼트인 Americas에서 다시 1부터 행 번호 지정이 시작됩니다.

NoOfRows() 열은 각 열 세그먼트의 행 수를 계산합니다. 예를 들어 유럽에는 열 세그먼트에 3개의 행이 있습니다.

NoOfRows()에 대한 TOTAL 인수로 인해 **NoOfRows(Total)** 열은 차원을 무시하고 테이블의 행을 계산합니다.

테이블이 두 번째 차원인 SubRegion을 기준으로 정렬된 경우 열 세그먼트는 해당 차원을 기반으로 하므로 각 SubRegion에 대해 행 번호가 변경됩니다.

관련 항목:

RowNo - 차트 함수 (page 563)

Peek

Peek()는 이미 로드된 행에 대한 테이블의 필드 값을 반환합니다. 테이블처럼 행 번호를 지정할 수 있습니다. 행 번호를 지정하지 않으면 이전에 마지막으로 로드된 레코드가 사용됩니다.

peek() 함수는 이전에 로드된 테이블, 즉 특정 필드의 첫 번째 값 또는 마지막 값에서 관련 경계를 찾는 데 가장 자주 사용됩니다. 대부분의 경우 이 값은 나중에 사용할 수 있도록 변수에 저장됩니다(예: do-while 루프의 조건).

구문:

```
Peek(
field_name
[, row_no[, table_name ] ])
```

반환 데이터 유형: dual

인수:

인수

인수	설명
field_name	반환 값이 필요한 필드의 이름입니다.입력 값은 문자열(예: 따옴표로 묶은 리터럴)로 지정해야 합니다.
row_no	테이블에서 필수 필드를 지정하는 행입니다. 표현식을 사용할 수 있지만 정수로 처리해야 합니다. 0은 첫 번째 레코드, 1은 두 번째 레코드와 같은 식으로 이어집니다. 음수는 테이블 끝을 기준으로 한 순서를 나타냅니다. 1은 마지막으로 읽은 레코드를 나타냅니다. **row_no**를 지정하지 않으면 -1이 사용됩니다.
table_name	끝 부분에 콜론이 없는 테이블 레이블. **table_name**을 지정하지 않으면 현재 테이블이 사용됩니다. **LOAD** 문 외부에서 사용하거나 다른 테이블을 참조할 경우 **table_name**을 반드시 포함해야 합니다.

제한 사항:

이 함수는 이미 로드된 레코드의 값만 반환할 수 있습니다. 즉, 테이블의 첫 번째 레코드에서 -1을 row_no로 사용하는 호출은 NULL을 반환합니다.

예 및 결과:

예 1

예제 스크립트를 앱에 추가하고 실행합니다. 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.

```
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;
```

결과 테이블

직원 코드	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. `Peek('EmployeeCode',0, 'EmployeeDates')`가 EmployeeDates 테이블의 EmployeeCode에 대한 첫 번째 값을 반환하기 때문입니다.

LastCode = 106. `Peek('EmployeeCode',-1, 'EmployeeDates')`가 EmployeeDates 테이블의 EmployeeCode에 대한 마지막 값을 반환하기 때문입니다.

다음과 같이 **row_no** 인수의 값을 바꾸면 테이블의 다른 행의 값을 반환합니다.

`Peek('EmployeeCode',2, 'EmployeeDates')`는 테이블의 세 번째 값(103)을 FirstCode로 반환합니다.

하지만 이 예에서 테이블을 세 번째 인수 **table_name**으로 지정하지 않으면 이 함수가 현재 테이블(이 경우 내부 테이블)을 참조합니다.

예 2

테이블에서 더 아래에 있는 데이터에 액세스하려면 두 단계로 수행해야 합니다. 먼저 전체 테이블을 임시 테이블에 로드한 다음 **Peek()**를 사용할 때 다시 정렬합니다.

예제 스크립트를 앱에 추가하고 실행합니다. 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.

```
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.

결과 테이블

ID	목록	값
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

IF() 문은 임시 테이블 T1에서 생성됩니다.
`Peek('ID')`는 현재 테이블 T2의 이전 행에서 ID 필드를 참조합니다.
`Peek('List')`는 표현식을 평가하면서 현재 생성 중인 테이블 T2의 이전 행에서 List 필드를 참조합니다.

이 문은 다음과 같이 평가됩니다.
ID의 현재 값이 ID의 이전 값과 동일한 경우, Value의 현재 값과 연결된 Peek('List')의 값을 작성합니다. 그렇지 않으면 Value의 현재 값만 작성합니다.

Peek('List')에 연결된 결과가 이미 있는 경우 Peek('List')의 새로운 결과가 연결됩니다.

***Order by** 절에 주의하십시오. 테이블 정렬 방법을 지정합니다(ID에 따라 오름차순으로). 이 절이 없으면 Peek() 함수가 내부 테이블에 지정된 임의의 순서를 사용하므로 예기치 못한 결과가 발생할 수 있습니다.*

예 3

예제 스크립트를 앱에 추가하고 실행합니다. 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.

```
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];
```

결과 테이블

Amount	AmountMonthBefore	월
1	4	2022-06
2	-	2022-01
3	2	2022-02
4	9	2022-05
7	3	2022-03
9	7	2022-04

AmountMonthBefore 필드에는 이전 달의 금액이 저장됩니다.

여기서 row_no 및 table_name 매개 변수는 생략되어 기본값이 사용됩니다. 이 예에서 다음 세 가지 함수 호출은 동일합니다.

- Peek(Amount)
- Peek(Amount,-1)
- Peek(Amount,-1,'Amounts')

row_no로 -1을 사용하면 이전 행의 값이 사용됩니다. 이 값을 대체하여 테이블의 다른 행 값을 가져올 수 있습니다.

Peek(Amount,2)는 테이블의 세 번째 값인 7을 반환합니다.

예 4

올바른 결과를 얻으려면 데이터를 올바르게 정렬해야 하지만 아쉽게도 항상 그런 것은 아닙니다. 또한 Peek() 함수는 아직 로드되지 않은 데이터를 참조하는 데 사용할 수 없습니다. 임시 테이블을 사용하고 데이터를 통해 여러 패스를 실행하면 이러한 문제를 피할 수 있습니다.

예제 스크립트를 앱에 추가하고 실행합니다. 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.

```
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;
```

설명

초기 테이블은 월별로 정렬됩니다. 즉, peek() 함수는 많은 경우 잘못된 제품에 대한 금액을 반환합니다. 따라서 이 테이블을 다시 정렬해야 합니다. 이는 새 테이블 tmp2Amounts를 만드는 데이터를 통해 두 번째 패스를 실행하여 수행됩니다. Order By 절에 주의하십시오. 먼저 제품별로 레코드를 정렬한 다음 월별로 오름차순으로 정렬합니다.

If() 함수는 AmountMonthBefore가 이전 행에 동일한 제품이지만 이전 달에 대한 데이터가 포함된 경우에만 계산되어야 하기 때문에 필요합니다. 현재 행의 제품과 이전 행의 제품을 비교하여 이 조건의 유효성을 검사할 수 있습니다.

두 번째 테이블이 만들어질 때 첫 번째 테이블 tmp1Amounts는 Drop Table 문을 사용하여 삭제됩니다.

마지막으로 세 번째 패스가 데이터를 통해 이루어지지만 이제는 월이 역순으로 정렬됩니다. 이런 식으로 AmountMonthAfter도 계산할 수 있습니다.

Order by 절은 테이블이 정렬되는 방식을 지정합니다. 이러한 절이 없으면 Peek() 함수는 내부 테이블에 있는 임의의 순서를 사용하므로 예측할 수 없는 결과가 발생할 수 있습니다.

결과

결과 테이블

월	제품	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	-

예 5

예제 스크립트를 앱에 추가하고 실행합니다. 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.

```
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;
```

결과

결과 테이블

분기	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

설명

LOAD 문 **Load *, rangesum(SumVal,peek('AccSumVal')) as AccSumVal**은 이전 값이 현재 값에 더해지는 재귀 호출을 포함합니다. 이 작업은 스크립트에서 값의 누적을 계산하는 데 사용됩니다.

관련 항목:

Previous

Previous()는 **where** 절로 인해 무시되지 않은 이전 입력 레코드의 데이터를 사용하여 **expr** 표현식의 값을 찾습니다. 이 함수는 내부 테이블의 첫 번째 레코드에 대해 NULL을 반환합니다.

구문:

```
Previous(expr)
```

반환 데이터 유형: dual

인수:

인수

인수	설명
expr	측정할 데이터가 포함된 표현식 또는 필드입니다. 레코드에 더 심층적으로 액세스하기 위해 표현식에 중첩된 **previous()** 함수를 포함할 수 있습니다. 입력 소스에서 데이터를 직접 가져오므로 관련 데이터베이스에 저장되지 않았더라도 Qlik Sense에 로드되지 않은 필드를 참조할 수 있습니다.

제한 사항:

이 함수는 내부 테이블의 첫 번째 레코드에 대해 NULL을 반환합니다.

로드 스크립트에 다음을 입력합니다.

```
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 '|');
```

Load 문에서 **Previous()** 함수를 사용하면 Sales의 현재 값을 이전 값과 비교하고 세 번째 필드인 Increase에서 사용할 수 있습니다.

결과 테이블

월	판매	증가
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 - 차트 함수

Top()은 테이블의 열 세그먼트 내에 있는 첫 번째(맨 위) 행의 표현식을 평가합니다. 계산되는 행은 **offset**(있는 경우) 값에 따라 달라지며 기본적으로는 맨 위에 있는 행입니다. 테이블 이외의 차트인 경우 **Top()**은 해당 차트의 일반표 동등 부분에 있는 현재 열의 첫 번째 행을 평가합니다.

구문:

```
Top([TOTAL] expr [ , offset [,count ]])
```

반환 데이터 유형: dual

인수:

인수

인수	설명
expr	측정할 데이터가 포함된 표현식 또는 필드입니다.
offset	n의 **offset**을 1보다 크게 지정하면 표현식의 평가가 맨 위 행에서 n 행 아래로 이동됩니다. 오프셋을 음수로 지정하면 **Top** 함수가 상응하는 양수 오프셋이 있는 **Bottom** 함수와 동일하게 작동합니다.

인수	설명
count	세 번째 파라메타 **count**를 1보다 크게 지정하면 이 함수에서 **count**의 범위 값을 반환하며, 현재 열 세그먼트의 마지막 **count** 행 각각에 대해 하나의 값을 반환합니다. 이러한 형식으로 이 함수를 특별 범위 함수의 인수로 활용할 수 있습니다. *범위 함수 (page 1282)*
TOTAL	테이블이 1차원이거나 **TOTAL** 한정자를 인수로 사용하는 경우 현재 열 세그먼트는 항상 전체 열과 동등합니다.

열 세그먼트는 현재 정렬 순서에서 차원에 대해 동일한 값을 갖는 셀의 연속된 하위 집합으로 정의됩니다. 인터 레코드 차트 함수는 동등한 일반표 차트의 가장 오른쪽 차원을 제외한 열 세그먼트에서 계산됩니다. 차트에 차원이 하나만 있는 경우 또는 TOTAL 한정자가 지정된 경우는 표현식이 전체 테이블을 평가합니다.

테이블 또는 테이블과 동등한 것에 여러 세로 차원이 있을 경우 현재 열 세그먼트에는 필드 간 정렬 순서에서 마지막 차원이 표시되는 열을 제외하고 모든 차원 열 내의 현재 행과 같은 값을 가진 행만 포함됩니다.

제한 사항:

- 재귀 호출은 NULL을 반환합니다.
- 이 차트 함수가 차트의 표현식에서 사용되는 경우 차트의 y 값에 대한 정렬 또는 테이블의 표현식 열에 의한 정렬은 허용되지 않습니다. 따라서 해당 정렬 옵션이 자동으로 비활성화됩니다. 시각화 또는 테이블에서 이 차트 함수를 사용하면 시각화의 정렬이 이 함수에 대해 정렬된 입력으로 되돌아갑니다.

예 및 결과:

1

이 예에서 보여 주는 테이블의 스크린샷에서, 테이블 시각화는 **Customer** 차원와 `Sum(Sales)` 측정값 및 `Top(Sum(Sales))`로부터 만들어집니다.

Top(Sum(Sales)) 열은 모든 행에 대해 맨 위 행 **Astrida**의 값인 587을 반환합니다.

이 테이블은 더 복잡한 계수를 보여줍니다. 하나는 `Sum(Sales)+Top(Sum(Sales))`에서 생성된 것이며, 하나는 레이블이 **Top offset 3**이고 표현식 `Sum(Sales)+Top(Sum(Sales), 3)`에서 생성된 것으로 인수 **offset**이 3으로 설정되었습니다. 이는 현재 행의 **Sum(Sales)** 값을 맨 위 행에서 세 번째 행의 값에 추가하는 것이며, 현재 행에 **Canutility**의 값을 더하는 것과 같습니다.

예 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

2

이 예에 표시된 테이블의 스크린샷에는 시각화에 더 많은 차원이 추가되었습니다. **Month** 및 **Product**의 측정값 2개를 사용합니다. 차원이 두 개 이상 있는 차트의 경우 **Above**, **Below**, **Top** 및 **Bottom** 함수를 포함하는 표현식의 결과는 Qlik Sense에서 열 차원이 정렬되는 순서에 따라 달라집니다. Qlik Sense에서는 차원의 마지막 정렬 결과인 열 세그먼트에 기반하여 해당 함수를 평가합니다. 열 정렬 순서는 속성 패널의 **정렬**에서 제어되며 이 순서가 반드시 테이블에 열이 표시되는 순서는 아닙니다.

예 2의 첫 번째 테이블입니다. Month(Jan)에 기반한 First value 측정값의 Top 값입니다.

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

예 2의 두 번째 테이블입니다. Product(Astrida의 AA)에 기반한 First value 측정값의 Top 값입니다.

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

자세한 내용은 **Above** 함수의 예: 2를 참조하십시오.

예 3

3	**결과**
Top 함수는 범위 함수에 대한 입력으로 사용할 수 있습니다. 합니다(예: `RangeAvg (Top(Sum (Sales),1,3))`).	**Top()** 함수에 대한 인수에서 offset는 1로 설정되고 count은 3으로 설정됩니다. 이 함수는 열 세그먼트에 있는 맨 아래 행 아래의 행에서 시작하는 세 행(offset=1이므로)과 그 아래의 두 행(행이 있는 곳)에 대한 표현식 **Sum(Sales)**의 결과를 찾습니다. 이 세 값은 제공된 숫자 범위에서 값의 평균을 구하는 RangeAvg() 함수에 대한 입력으로 사용됩니다. 차원으로 **Customer**가 포함된 테이블은 RangeAvg() 표현식에 대해 다음과 같은 결과를 제공합니다.
	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 '|');
```

관련 항목:

- *Bottom - 차트 함수 (page 1236)*
- *Above - 차트 함수 (page 1228)*
- *Sum - 차트 함수 (page 338)*
- *RangeAvg (page 1284)*
- *범위 함수 (page 1282)*

SecondaryDimensionality - 차트 함수

SecondaryDimensionality()는 비집계 내용이 있는, 즉 부분합 또는 축소된 집계가 포함되지 않은 차원 피벗 테이블 행의 수를 반환합니다. 이 함수는 가로 피벗 테이블 차원에 해당하는 **dimensionality()** 함수와 동등합니다.

구문:

```
SecondaryDimensionality( )
```

반환 데이터 유형: 정수

제한 사항:

- 피벗 테이블에서 사용되지 않는 경우 **SecondaryDimensionality** 함수는 항상 0을 반환합니다.
- 이 차트 함수가 차트의 표현식에서 사용되는 경우 차트의 y 값에 대한 정렬 또는 테이블의 표현식 열에 의한 정렬은 허용되지 않습니다. 따라서 해당 정렬 옵션이 자동으로 비활성화됩니다. 시각화 또는 테이블에서 이 차트 함수를 사용하면 시각화의 정렬이 이 함수에 대해 정렬된 입력으로 되돌아갑니다.

After - 차트 함수

After()는 피벗 테이블의 차원 값이 피벗 테이블의 행 세그먼트 내 현재 열 뒤의 열에 표시되므로 피벗 테이블의 차원 값으로 평가된 표현식의 값을 반환합니다.

구문:

```
after([TOTAL] expr [, offset [, count ]])
```

이 차트 함수가 차트의 표현식에서 사용되는 경우 차트의 y 값에 대한 정렬 또는 테이블의 표현식 열에 의한 정렬은 허용되지 않습니다. 따라서 해당 정렬 옵션이 자동으로 비활성화됩니다. 시각화 또는 테이블에서 이 차트 함수를 사용하면 시각화의 정렬이 이 함수에 대해 정렬된 입력으로 되돌아갑니다.

이 함수는 피벗 테이블을 제외한 모든 차트 유형에서 NULL을 반환합니다.

인수:

인수	설명
expr	측정할 데이터가 포함된 표현식 또는 필드입니다.
offset	**offset** n을 1보다 크게 지정하면 표현식의 평가가 현재 행에서 n개 행 오른쪽으로 이동합니다. 오프셋을 0으로 지정하면 현재 행의 표현식이 평가됩니다. 오프셋을 음수로 지정하면 **After** 함수가 상응하는 양수 오프셋이 있는 **Before** 함수와 동일하게 작동합니다.
count	세 번째 파라메타 **count**를 1보다 크게 지정하면, 함수는 값의 범위를 반환하며, 원래 셀에서 오른쪽으로 계산하여 **count** 값까지 테이블 행 각각에 대해 하나의 값이 반환됩니다.
TOTAL	테이블이 1차원이거나 **TOTAL** 한정자를 인수로 사용하는 경우 현재 열 세그먼트는 항상 전체 열과 동등합니다.

행 세그먼트의 마지막 열 아래에는 다른 열이 없으므로 NULL 값이 반환됩니다.

피벗 테이블에 여러 가로 차원이 있을 경우 현재 행 세그먼트에는 필드 간 정렬 순서에서 마지막 가로 차원이 표시되는 행을 제외하고 모든 차원 행 내의 현재 열과 같은 값을 가진 열만 포함됩니다. 피벗 테이블에서 가로 차원의 필드 간 정렬 순서는 간단하게 위쪽에서 아래쪽의 차원 순서로 정의됩니다..

```
after( sum( Sales ))
after( sum( Sales ), 2 )
after( total sum( Sales ))
```

`rangeavg (after(sum(x),1,3))`는 현재 열 바로 오른쪽에 있는 열 3개에서 평가한 **sum(x)** 함수의 결과 3개의 평균을 반환합니다.

Before - 차트 함수

Before()는 피벗 테이블의 차원 값이 피벗 테이블의 행 세그먼트 내 현재 열 앞의 열에 표시되므로 피벗 테이블의 차원 값으로 평가된 표현식의 값을 반환합니다.

구문:

```
before([TOTAL] expr [, offset [, count]])
```

이 함수는 피벗 테이블을 제외한 모든 차트 유형에서 NULL을 반환합니다.

이 차트 함수가 차트의 표현식에서 사용되는 경우 차트의 y 값에 대한 정렬 또는 테이블의 표현식 열에 의한 정렬은 허용되지 않습니다. 따라서 해당 정렬 옵션이 자동으로 비활성화됩니다. 시각화 또는 테이블에서 이 차트 함수를 사용하면 시각화의 정렬이 이 함수에 대해 정렬된 입력으로 되돌아갑니다.

인수:

인수

인수	설명
expr	측정할 데이터가 포함된 표현식 또는 필드입니다.
offset	**offset** n,을 1보다 크게 지정하면 표현식의 평가가 현재 행에서 n개 행 왼쪽으로 이동합니다. 오프셋을 0으로 지정하면 현재 행의 표현식이 평가됩니다. 오프셋을 음수로 지정하면 **Before** 함수가 상응하는 양수 오프셋이 있는 **After** 함수와 동일하게 작동합니다.
count	세 번째 파라메타 **count**를 1보다 크게 지정하면, 함수는 값의 범위를 반환하며, 원래 셀에서 왼쪽으로 계산하여 **count** 값까지 테이블 행 각각에 대해 하나의 값이 반환됩니다.
TOTAL	테이블이 1차원이거나 **TOTAL** 한정자를 인수로 사용하는 경우 현재 열 세그먼트는 항상 전체 열과 동등합니다.

행 세그먼트의 첫 번째 열 위에는 다른 열이 없으므로 NULL 값이 반환됩니다.

피벗 테이블에 여러 가로 차원이 있을 경우 현재 행 세그먼트에는 필드 간 정렬 순서에서 마지막 가로 차원이 표시되는 행을 제외하고 모든 차원 행 내의 현재 열과 같은 값을 가진 열만 포함됩니다.피벗 테이블에서 가로 차원의 필드 간 정렬 순서는 간단하게 위쪽에서 아래쪽의 차원 순서로 정의됩니다..

```
before( sum( Sales ))
before( sum( Sales ), 2 )
before( total sum( Sales ))
```

`rangeavg (before(sum(x),1,3))`는 현재 열 바로 왼쪽에 있는 열 3개에서 평가한 **sum(x)** 함수의 결과 3개의 평균을 반환합니다.

First - 차트 함수

First()는 피벗 테이블의 현재 행 세그먼트의 첫 번째 열에 표시되는 피벗 테이블의 차원 값으로 평가된 표현식의 값을 반환합니다. 이 함수는 피벗 테이블을 제외한 모든 차트 유형에서 NULL을 반환합니다.

이 차트 함수가 차트의 표현식에서 사용되는 경우 차트의 y 값에 대한 정렬 또는 테이블의 표현식 열에 의한 정렬은 허용되지 않습니다. 따라서 해당 정렬 옵션이 자동으로 비활성화됩니다. 시각화 또는 테이블에서 이 차트 함수를 사용하면 시각화의 정렬이 이 함수에 대해 정렬된 입력으로 되돌아갑니다.

구문:

```
first([TOTAL] expr [, offset [, count]])
```

인수:

인수

인수	설명
expression	측정할 데이터가 포함된 표현식 또는 필드입니다.
offset	**offset** n을 1보다 크게 지정하면 표현식의 평가가 현재 행에서 n개 행 오른쪽으로 이동합니다. 오프셋을 0으로 지정하면 현재 행의 표현식이 평가됩니다. 오프셋을 음수로 지정하면 **First** 함수가 상응하는 양수 오프셋이 있는 **Last** 함수와 동일하게 작동합니다.
count	세 번째 파라메타 **count**를 1보다 크게 지정하면, 함수는 값의 범위를 반환하며, 원래 셀에서 오른쪽으로 계산하여 **count** 값까지 테이블 행 각각에 대해 하나의 값이 반환됩니다.
TOTAL	테이블이 1차원이거나 **TOTAL** 한정자를 인수로 사용하는 경우 현재 열 세그먼트는 항상 전체 열과 동등합니다.

피벗 테이블에 여러 가로 차원이 있을 경우 현재 행 세그먼트에는 필드 간 정렬 순서에서 마지막 가로 차원이 표시되는 행을 제외하고 모든 차원 행 내의 현재 열과 같은 값을 가진 열만 포함됩니다.피벗 테이블에서 가로 차원의 필드 간 정렬 순서는 간단하게 위쪽에서 아래쪽의 차원 순서로 정의됩니다..

```
first( sum( Sales ))
first( sum( Sales ), 2 )
first( total sum( Sales )
```

`rangeavg (first(sum(x),1,5))`는 현재 행 세그먼트의 가장 왼쪽 열 5개에서 평가한 **sum(x)** 함수 결과의 평균을 반환합니다.

Last - 차트 함수

Last()는 피벗 테이블의 현재 행 세그먼트의 마지막 열에 표시되는 피벗 테이블의 차원 값으로 평가된 표현식의 값을 반환합니다. 이 함수는 피벗 테이블을 제외한 모든 차트 유형에서 NULL을 반환합니다.

이 차트 함수가 차트의 표현식에서 사용되는 경우 차트의 y 값에 대한 정렬 또는 테이블의 표현식 열에 의한 정렬은 허용되지 않습니다. 따라서 해당 정렬 옵션이 자동으로 비활성화됩니다. 시각화 또는 테이블에서 이 차트 함수를 사용하면 시각화의 정렬이 이 함수에 대해 정렬된 입력으로 되돌아갑니다.

구문:

```
last([TOTAL] expr [, offset [, count]])
```

인수:

인수

인수	설명
expr	측정할 데이터가 포함된 표현식 또는 필드입니다.
offset	**offset** n,을 1보다 크게 지정하면 표현식의 평가가 현재 행에서 n개 행 왼쪽으로 이동합니다. 오프셋을 0으로 지정하면 현재 행의 표현식이 평가됩니다. 오프셋을 음수로 지정하면 **First** 함수가 상응하는 양수 오프셋이 있는 **Last** 함수와 동일하게 작동합니다.
count	세 번째 파라메타 **count**를 1보다 크게 지정하면, 함수는 값의 범위를 반환하며, 원래 셀에서 왼쪽으로 계산하여 **count** 값까지 테이블 행 각각에 대해 하나의 값이 반환됩니다.
TOTAL	테이블이 1차원이거나 **TOTAL** 한정자를 인수로 사용하는 경우 현재 열 세그먼트는 항상 전체 열과 동등합니다.

피벗 테이블에 여러 가로 차원이 있을 경우 현재 행 세그먼트에는 필드 간 정렬 순서에서 마지막 가로 차원이 표시되는 행을 제외하고 모든 차원 행 내의 현재 열과 같은 값을 가진 열만 포함됩니다.피벗 테이블에서 가로 차원의 필드 간 정렬 순서는 간단하게 위쪽에서 아래쪽의 차원 순서로 정의됩니다..

```
last( sum( Sales ))
last( sum( Sales ), 2 )
last( total sum( Sales )
```

`rangeavg (last(sum(x),1,5))`는 현재 행 세그먼트의 가장 오른쪽 열 5개에서 평가한 **sum(x)** 함수 결과의 평균을 반환합니다.

ColumnNo - 차트 함수

ColumnNo()는 피벗 테이블의 현재 행 세그먼트 내에 있는 현재 열의 번호를 반환합니다. 첫 번째 열은 1번입니다.

구문:

```
ColumnNo([total])
```

인수:

인수

인수	설명
TOTAL	테이블이 1차원이거나 **TOTAL** 한정자를 인수로 사용하는 경우 현재 열 세그먼트는 항상 전체 열과 동등합니다.

피벗 테이블에 여러 가로 차원이 있을 경우 현재 행 세그먼트에는 필드 간 정렬 순서에서 마지막 가로 차원이 표시되는 행을 제외하고 모든 차원 행 내의 현재 열과 같은 값을 가진 열만 포함됩니다.피벗 테이블에서 가로 차원의 필드 간 정렬 순서는 간단하게 위쪽에서 아래쪽의 차원 순서로 정의됩니다..

이 차트 함수가 차트의 표현식에서 사용되는 경우 차트의 y 값에 대한 정렬 또는 테이블의 표현식 열에 의한 정렬은 허용되지 않습니다. 따라서 해당 정렬 옵션이 자동으로 비활성화됩니다. 시각화 또는 테이블에서 이 차트 함수를 사용하면 시각화의 정렬이 이 함수에 대해 정렬된 입력으로 되돌아갑니다.

```
if( ColumnNo( )=1, 0, sum( Sales ) / before( sum( Sales )))
```

NoOfColumns - 차트 함수

NoOfColumns()는 피벗 테이블 내 현재 행 세그먼트에 있는 열의 수를 반환합니다.

이 차트 함수가 차트의 표현식에서 사용되는 경우 차트의 y 값에 대한 정렬 또는 테이블의 표현식 열에 의한 정렬은 허용되지 않습니다. 따라서 해당 정렬 옵션이 자동으로 비활성화됩니다. 시각화 또는 테이블에서 이 차트 함수를 사용하면 시각화의 정렬이 이 함수에 대해 정렬된 입력으로 되돌아갑니다.

구문:

```
NoOfColumns([total])
```

인수:

인수

인수	설명
TOTAL	테이블이 1차원이거나 **TOTAL** 한정자를 인수로 사용하는 경우 현재 열 세그먼트는 항상 전체 열과 동등합니다.

피벗 테이블에 여러 가로 차원이 있을 경우 현재 행 세그먼트에는 필드 간 정렬 순서에서 마지막 차원이 표시되는 행을 제외하고 모든 차원 행 내의 현재 열과 같은 값을 가진 열만 포함됩니다.피벗 테이블에서 가로 차원의 필드 간 정렬 순서는 간단하게 위쪽에서 아래쪽의 차원 순서로 정의됩니다..

```
if( ColumnNo( )=NoOfColumns( ), 0, after( sum( Sales )))
```

8.17 논리 함수

이 섹션에서는 논리 연산을 처리하는 함수에 대해 설명합니다. 모든 함수는 데이터 로드 스크립트와 차트 표현식 모두에서 사용할 수 있습니다.

IsNum
표현식을 숫자로 해석할 수 있으면 -1(True)을, 그렇지 않으면 0(False)을 반환합니다.

```
IsNum( expr )
```

IsText
표현식에 텍스트 표현이 있으면 -1(True)을, 그렇지 않으면 0(False)을 반환합니다.

```
IsText( expr )
```

*표현식이 NULL인 경우 **IsNum** 및 **IsText**는 모두 0을 반환합니다.*

다음 예에서는 텍스트와 숫자 값이 혼합된 인라인 테이블을 로드하고 값이 숫자 값인지,텍스트 값인지 각각 확인하는 두 필드를 추가합니다.

```
Load *, IsNum(Value), IsText(Value)
Inline [
Value
23
Green
Blue
12
33Red];
```

결과 테이블은 다음과 같습니다.

Resulting table

Value	IsNum(Value)	IsText(Value)
23	-1	0
Green	0	-1
Blue	0	-1
12	-1	0
33Red	0	-1

8.18 매핑 함수

이 섹션에서는 매핑 테이블을 처리하는 함수에 대해 설명합니다. 매핑 테이블은 스크립트 실행 중에 필드 값 또는 필드 이름을 바꾸는 데 사용할 수 있습니다.

매핑 함수는 데이터 로드 스크립트에서만 사용할 수 있습니다.

매핑 함수 개요

각 함수는 개요가 끝난 후에 더 자세히 설명합니다. 구문에서 함수 이름을 클릭하여 해당 함수에 대한 상세 설명에 즉시 액세스할 수도 있습니다.

ApplyMap
ApplyMap 스크립트 함수는 이전에 로드된 매핑 테이블에 표현식의 결과를 매핑하는 데 사용됩니다.

```
ApplyMap ('mapname', expr [ , defaultexpr ] )
```

MapSubstring
MapSubstring 스크립트 함수는 이전에 로드된 매핑 테이블에 표현식의 일부를 매핑하는 데 사용됩니다. 매핑은 대/소문자가 구분되고 반복적이지 않으며, 부분 문자열은 왼쪽에서 오른쪽으로 매핑됩니다.

```
MapSubstring ('mapname', expr)
```

ApplyMap

ApplyMap 스크립트 함수는 이전에 로드된 매핑 테이블에 표현식의 결과를 매핑하는 데 사용됩니다.

구문:

```
ApplyMap('map_name', expression [ , default_mapping ] )
```

반환 데이터 유형: dual

인수:

인수

인수	설명
map_name	이전에 **mapping load** 또는 **mapping select** 문을 통해 생성된 매핑 테이블의 이름입니다. 이 이름은 작은따옴표로 묶어야 합니다. *매크로 확장 변수에서 이 함수를 사용하고 존재하지 않는 매핑 테이블을 참조하면 함수 호출에 실패하고 필드가 만들어지지 않습니다.*
expression	결과가 매핑될 표현식입니다.
default_ mapping	이 인수를 지정하면 매핑 테이블에 expression에 일치하는 값이 없을 경우 이 값이 기본값으로 사용됩니다. 지정하지 않으면 expression의 값이 그대로 반환됩니다.

ApplyMap의 출력 필드는 입력 필드의 이름과 동일한 이름을 가질 수 없습니다. 이 경우 예기치 않은 결과가 발생할 수 있습니다. 사용하지 않는 예: `ApplyMap('Map', A) as A`.

이 예에서는 거주 국가를 나타내는 국가 코드에 따라 영업직원 목록을 로드합니다. 국가 코드와 국가를 매핑하는 테이블을 사용하여 국가 코드를 국가명으로 바꿉니다. 매핑 테이블에는 3개 국가만 정의되어 있으며 다른 국가 코드는 'Rest of the world'로 매핑됩니다.

```
// 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';
```

결과 테이블(Salespersons)은 다음과 같습니다.

Resulting table

Salesperson	**Country**
John	Sweden
Mary	Sweden
Per	Sweden
Preben	Denmark
Olle	Denmark

Salesperson	Country
Ole	Norway
Risttu	Rest of the world

MapSubstring

MapSubstring 스크립트 함수는 이전에 로드된 매핑 테이블에 표현식의 일부를 매핑하는 데 사용됩니다. 매핑은 대/소문자가 구분되고 반복적이지 않으며, 부분 문자열은 왼쪽에서 오른쪽으로 매핑됩니다.

구문:

```
MapSubstring('map_name', expression)
```

반환 데이터 유형: 문자열

인수:

인수

인수	설명
map_name	이전에 **mapping load** 또는 **mapping select** 문으로 읽은 매핑 테이블의 이름입니다. 이 이름은 곧은 작은따옴표로 묶어야 합니다.  *매크로 확장 변수에서 이 함수를 사용하고 존재하지 않는 매핑 테이블을 참조하면 함수 호출에 실패하고 필드가 만들어지지 않습니다.*
expression	부분 문자열이 결과를 매핑할 표현식입니다.

이 예에서는 제품 모델 목록을 로드합니다. 각 모델에는 복합 코드로 지정된 일련의 특성이 있습니다. 매핑 테이블을 MapSubstring과 함께 사용하면 특성 코드를 설명으로 확장할 수 있습니다.

```
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';
```

결과 테이블은 다음과 같습니다.

Resulting table

Model	Description
Twixie	Red Cotton Small
Boomer	Blue Polyester Large
Raven	Yellow Polyester Medium
Seedling	Red Cotton Large
SeedlingPlus	Red Cotton Large with hood
Younger	Blue Cotton with patch
MultiStripe	Red Yellow Blue Cotton Small/Medium/Large

8.19 수학 함수

이 섹션에서는 수학적 상수와 부울 값에 대한 함수를 설명합니다. 이러한 함수에는 어떠한 매개 변수도 없지만 괄호는 여전히 필요합니다.

모든 함수는 데이터 로드 스크립트와 차트 표현식 모두에서 사용할 수 있습니다.

e
이 함수는 자연 로그 **e**(2.71828...)의 기수를 반환합니다.

```
e( )
```

false
이 함수는 표현식에서 논리적 거짓으로 사용할 수 있는 이중 값(텍스트 값 'False' 및 숫자 값 0 포함)을 반환합니다.

```
false( )
```

pi
이 함수는 π(3.14159...) 값을 반환합니다.

```
pi( )
```

rand
이 함수는 0과 1 사이의 임의의 숫자를 반환합니다. 이는 샘플 데이터를 만드는 데 사용할 수 있습니다.

```
rand( )
```

이 예제 스크립트는 무작위로 선택된 대문자(65 ~ 91 범위(65+26))가 포함된 1000개의 레코드로 구성된 테이블을 만듭니다.

```
Load
     Chr( Floor(rand() * 26) + 65) as UCaseChar,
     RecNo() as ID
     Autogenerate 1000;
```

true
이 함수는 표현식에서 논리적 참으로 사용할 수 있는 이중 값(텍스트 값 'True' 및 숫자 값 -1 포함)을 반환합니다.

```
true( )
```

8.20 NULL 함수

이 섹션에서는 NULL 값을 반환하거나 감지하는 함수에 대해 설명합니다.

모든 함수는 데이터 로드 스크립트와 차트 표현식 모두에서 사용할 수 있습니다.

NULL 함수 개요

각 함수는 개요가 끝난 후에 더 자세히 설명합니다. 구문에서 함수 이름을 클릭하여 해당 함수에 대한 상세 설명에 즉시 액세스할 수도 있습니다.

EmptyIsNull
EmptyIsNull 함수는 빈 문자열을 NULL로 변환합니다. 그러므로 매개변수가 빈 문자열이면 NULL을 반환하고 그렇지 않으면 매개변수를 반환합니다.

```
EmptyIsNull (expr )
```

IsNull
IsNull 함수는 표현식의 값이 NULL인지 테스트하여, 그런 경우 -1(True)을 반환하고 그렇지 않으면 0(False)을 반환합니다.

```
IsNull (expr )
```

Null
Null 함수는 NULL 값을 반환합니다.

```
NULL( )
```

EmptyIsNull

EmptyIsNull 함수는 빈 문자열을 NULL로 변환합니다. 그러므로 매개변수가 빈 문자열이면 NULL을 반환하고 그렇지 않으면 매개변수를 반환합니다.

구문:

```
EmptyIsNull(exp )
```

예 및 결과:

스크립팅 예

예	결과
`EmptyIsNull(AdditionalComments)`	이 표현식은 빈 문자열 대신 *AdditionalComments* 필드의 빈 문자열 값을 null로 반환합니다. 비어 있지 않은 문자열과 숫자가 반환됩니다.
`EmptyIsNull(PurgeChar (PhoneNumber, ' -()'))`	이 표현식은 *PhoneNumber* 필드에서 대시, 공백 및 괄호를 제거합니다. 남은 문자가 없으면 EmptyIsNull 함수는 빈 문자열을 null로 반환합니다. 빈 전화 번호는 전화 번호가 없는 것과 같습니다.

IsNull

IsNull 함수는 표현식의 값이 NULL인지 테스트하여, 그런 경우 -1(True)을 반환하고 그렇지 않으면 0(False)을 반환합니다.

구문:

```
IsNull(expr )
```

*길이가 0인 문자열을 NULL로 처리하지 않으며 **IsNull**에서는 False를 반환합니다.*

데이터 로드 스크립트

이 예에서는 4행의 인라인 테이블이 로드되며, 처음 3행은 - 열에 아무것도 없거나 'NULL' 또는 Value이 포함되어 있습니다. **Null** 함수를 사용하여 중간 선행 **LOAD**를 통해 이들 값을 실제 NULL 값으로 변환할 수 있습니다.

첫 번째 선행 **LOAD**는 **IsNull** 함수를 사용하여 값이 NULL인지 확인하는 필드를 추가합니다.

```
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];
```

결과 테이블은 다음과 같습니다. ValueNullConv 열에 NULL 값이 -으로 표시되어 있습니다.

Resulting table

ID	Value	ValueNullConv	IsItNull
0		-	T
1	NULL	-	T
2	-	-	T
3	Value	Value	F

NULL

Null 함수는 NULL 값을 반환합니다.

구문:

```
Null( )
```

데이터 로드 스크립트

이 예에서는 4행의 인라인 테이블이 로드되며, 처음 3행은 - 열에 아무것도 없거나 'NULL' 또는 Value이 포함되어 있습니다. 이러한 값을 실제 NULL 값 표현으로 변환하려고 합니다.

중간의 선행 **LOAD**가 **Null** 함수를 사용하여 변환을 수행합니다.

첫 번째 선행 **LOAD**는 값이 NULL인지 확인하는 필드를 추가하며, 이 예에서 예시용으로 사용됩니다.

```
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];
```

결과 테이블은 다음과 같습니다. ValueNullConv 열에 NULL 값이 -으로 표시되어 있습니다.

Resulting table

ID	Value	ValueNullConv	IsItNull
0		-	T
1	NULL	-	T
2	-	-	T
3	Value	Value	F

8.21 범위 함수

범위 함수는 값의 배열을 받아 단일 값을 결과로 산출하는 함수입니다. 모든 범위 함수는 데이터 로드 스크립트와 차트 표현식 모두에서 사용할 수 있습니다.

예를 들어, 시각화에서 범위 함수를 사용하여 인터 레코드 배열로부터 단일 값을 계산할 수 있습니다. 또는 데이터 로드 스크립트에서 범위 함수를 사용하여 내부 테이블에 있는 값 배열로부터 단일 값을 계산할 수 있습니다.

범위 함수는 더 이상 사용하지 않는 일반 숫자 함수 ***numsum****,* ***numavg****,* ***numcount****,* ***nummin*** *및* ***nummax****를 대체합니다.*

기본 범위 함수

RangeMax

RangeMax()는 표현식 또는 필드 내에서 찾은 가장 높은 숫자 값을 반환합니다.

```
RangeMax (first_expr[, Expression])
```

RangeMaxString

RangeMaxString()은 표현식 또는 필드에서 찾은 텍스트 정렬 순서의 마지막 값을 반환합니다.

```
RangeMaxString (first_expr[, Expression])
```

RangeMin

RangeMin()은 표현식 또는 필드 내에서 찾은 가장 낮은 숫자 값을 반환합니다.

```
RangeMin (first_expr[, Expression])
```

RangeMinString

RangeMinString()은 표현식 또는 필드에서 찾은 텍스트 정렬 순서의 첫 번째 값을 반환합니다.

```
RangeMinString (first_expr[, Expression])
```

RangeMode

RangeMode()는 표현식 또는 필드에서 가장 발생 빈도가 높은 값(모드 값)을 찾습니다.

```
RangeMode (first_expr[, Expression])
```

RangeOnly

RangeOnly()는 표현식이 하나의 고유한 값으로 평가되는 경우 값을 반환하는 dual 함수입니다. 해당되지 않는 경우에는 **NULL**을 반환합니다.

```
RangeOnly (first_expr[, Expression])
```

RangeSum

RangeSum()은 값 범위의 합계를 반환합니다. 숫자가 아닌 모든 값은 0으로 처리됩니다.

```
RangeSum (first_expr[, Expression])
```

카운터 범위 함수

RangeCount

RangeCount()는 표현식 또는 필드에 있는 텍스트 및 숫자 값의 수를 반환합니다.

```
RangeCount (first_expr[, Expression])
```

RangeMissingCount

RangeMissingCount()는 표현식 또는 필드에 있는 숫자 이외의 값(NULL 포함)의 수를 반환합니다.

```
RangeMissingCount (first_expr[, Expression])
```

RangeNullCount

RangeNullCount()는 표현식 또는 필드에서 NULL 값의 수를 찾습니다.

```
RangeNullCount (first_expr[, Expression])
```

RangeNumericCount

RangeNumericCount()는 표현식 또는 필드에서 숫자 값의 수를 찾습니다.

```
RangeNumericCount (first_expr[, Expression])
```

RangeTextCount

RangeTextCount()는 표현식 또는 필드에 있는 텍스트 값의 수를 반환합니다.

```
RangeTextCount (first_expr[, Expression])
```

통계 범위 함수

RangeAvg

RangeAvg()는 범위의 평균을 반환합니다. 이 함수의 입력은 값의 범위 또는 표현식이 될 수 있습니다.

```
RangeAvg (first_expr[, Expression])
```

RangeCorrel

RangeCorrel()은 두 데이터 집합에 대한 상관 계수를 반환합니다. 상관 계수는 데이터 셋 간의 관계에 대한 측정값입니다.

```
RangeCorrel (x_values , y_values[, Expression])
```

RangeFractile

RangeFractile()은 숫자 범위의 n번째 **fractile**(사분위수)에 해당하는 값을 반환합니다.

```
RangeFractile (fractile, first_expr[ ,Expression])
```

RangeKurtosis

RangeKurtosis()는 숫자 범위의 첨도에 해당하는 값을 반환합니다.

```
RangeKurtosis (first_expr[, Expression])
```

RangeSkew

RangeSkew()는 숫자 범위의 왜곡도에 해당하는 값을 반환합니다.

```
RangeSkew (first_expr[, Expression])
```

RangeStdev

RangeStdev()는 숫자 범위의 표준 편차를 찾습니다.

```
RangeStdev (expr1[, Expression])
```

재무 범위 함수

RangeIRR

RangeIRR()은 입력 값으로 표현된 일련의 현금 흐름에 대한 내부 수익률을 반환합니다.

```
RangeIRR (value[, value][, Expression])
```

RangeNPV

RangeNPV()는 할인율 및 일련의 주기적인 미래 납입금(음수 값)과 수입(양수 값)을 기준으로 한 투자의 순 현재 가치를 반환합니다. 결과의 기본 숫자 서식은 **money**입니다.

```
RangeNPV (discount_rate, value[, value][, Expression])
```

RangeXIRR

RangeXIRR()은 주기적일 필요가 없는 현금 흐름의 일정에 대한 내부 수익률(연간)을 반환합니다. 일련의 주기적인 현금 흐름에 대한 내부 수익률을 계산하려면 **RangeIRR** 함수를 사용하십시오.

```
RangeXIRR (values, dates[, Expression])
```

RangeXNPV

RangeXNPV()는 **pmt** 및 **date**로 지정된 표현식에서 숫자 쌍으로 표현된 현금 흐름 일정(반드시 주기적일 필요는 없음)의 순 현재 가치를 반환합니다. 모든 납입금은 1년 365일을 기준으로 할인됩니다.

```
RangeXNPV (discount_rate, values, dates[, Expression])
```

관련 항목:

인터 레코드 함수 (page 1225)

RangeAvg

RangeAvg()는 범위의 평균을 반환합니다. 이 함수의 입력은 값의 범위 또는 표현식이 될 수 있습니다.

구문:

```
RangeAvg(first_expr[, Expression])
```

반환 데이터 유형: 숫자

인수:

이 함수의 인수에는 자체적으로 값 목록을 반환하는 인터 레코드 함수가 포함될 수 있습니다.

인수

인수	설명
first_expr	측정할 데이터가 포함된 표현식 또는 필드입니다.
Expression	측정할 데이터 범위가 포함된 옵션 표현식 또는 필드입니다.

제한 사항:

숫자 값이 발견되지 않으면 NULL이 반환됩니다.

예 및 결과:

스크립팅 예

예	결과
`RangeAvg (1,2,4)`	2.33333333을 반환합니다.
`RangeAvg (1,'xyz')`	1을 반환합니다.
`RangeAvg (null( ), 'abc')`	NULL를 반환합니다.

예제 스크립트를 앱에 추가하고 실행합니다. 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.

```
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
];
```

결과 테이블에 테이블 내 각 레코드에 대한 MyRangeAvg의 반환된 값이 표시됩니다.

결과 테이블

RangeID	MyRangeAvg
1	7

RangeID	MyRangeAvg
2	4
3	6
4	12.666
5	6.333
6	5

표현식이 포함된 예:

```
RangeAvg (Above(MyField),0,3))
```

현재 행과 현재 행 위의 두 행에서 계산된 **MyField**의 세 값 범위의 결과의 가변 평균을 반환합니다. 세 번째 인수를 3으로 지정하면, 위에 충분한 행이 있을 경우 **Above()** 함수에서 세 값을 반환하며, 이는 **RangeAvg()** 함수에 대한 입력으로 사용됩니다.

예에서 사용된 데이터:

*예상대로 예제가 작동하도록 **MyField**의 정렬을 비활성화합니다.*

샘플 데이터

MyField	RangeAvg (Above(MyField,0,3))	Comments
10	10	이는 맨 위 행이므로 범위에 하나의 값만이 포함됩니다.
2	6	이 행 위에는 하나의 행뿐이므로 범위는 10,2입니다.
8	6.6666666667	RangeAvg(10,2,8)와 동등
18	9.333333333	-
5	10. 333333333	-
9	10.6666666667	-

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
] ;
```

관련 항목:

- *Avg - 차트 함수 (page 389)*
- *Count - 차트 함수 (page 342)*

RangeCorrel

RangeCorrel()은 두 데이터 집합에 대한 상관 계수를 반환합니다. 상관 계수는 데이터 셋 간의 관계에 대한 측정값입니다.

구문:

```
RangeCorrel(x_value , y_value[, Expression])
```

반환 데이터 유형: 숫자

데이터 수열은 (x,y) 쌍으로 입력해야 합니다. 예를 들어, 배열 1이 2,6,9이고 배열 2가 3,8,4일 때 배열 1과 배열 2의 두 데이터 수열을 평가하려면 `RangeCorrel (2,3,6,8,9,4)`라고 입력하며, 이 경우 0.269가 반환됩니다.

인수:

인수

인수	설명
x-value, y-value	각 값은 선택적인 세 번째 파라메타가 포함된 인터 레코드 함수에서 반환된 단일 값 또는 값의 범위입니다. 각각의 값 또는 값의 범위는 **x-value** 또는 **y-values**의 범위에 대응해야 합니다.
Expression	측정할 데이터 범위가 포함된 옵션 표현식 또는 필드입니다.

제한 사항:

이 함수를 계산하려면 좌표 쌍이 최소한 2개 이상 필요합니다.

텍스트 값, NULL 값 및 누락된 값은 NULL을 반환합니다.

예 및 결과:

함수 예

예	결과
`RangeCorrel (2,3,6,8,9,4,8,5)`	0.2492를 반환합니다. 이 함수는 스크립트에서 로드되거나 식 편집기에서 시각화에 추가될 수 있습니다.

예제 스크립트를 앱에 추가하고 실행합니다. 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.

```
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 '|');
```

ID1가 차원이고 측정값이 RangeCorrel(x1,y1,x2,y2,x3,y3,x4,y4,x5,y5,x6,y6))인 테이블에서 **RangeCorrel()** 함수는 각 ID1 값에 대해 6개의 x,y 쌍 범위에서 **Correl** 값을 찾습니다.

결과 테이블

ID1	**MyRangeCorrel**
01	-0.9517
02	-0.5209
03	-0.5209
04	-0.1599

```
XY:
LOAD recno() as RangeID, * Inline [
X|Y
2|3
6|8
9|4
8|5
](delimiter is '|');
```

RangeID을 차원 및 측정값으로 하는 테이블: RangeCorrel(Below(X,0,4,BelowY,0,4)), **RangeCorrel()** 함수는 세 번째 인수 (count)가 4로 설정되어 로드된 테이블 XY에서 네 개의 x-y 값 범위를 생성하는 **Below()** 함수의 결과를 사용합니다.

결과 테이블

RangeID	**MyRangeCorrel2**
01	0.2492
02	-0.9959
03	-1.0000
04	-

RangeID 01의 값은 수동으로 입력한 RangeCorrel(2,3,6,8,9,4,8,5)와 동일합니다. RangeID의 다른 값의 경우 Below() 함수에 의해 생성된 수열은 다음과 같습니다. (6,8,9,4,8,5), (9,4,8,5) 및 (8,5), 마지막에 null 결과가 생성됩니다.

관련 항목:

Correl - 차트 함수 (page 392)

RangeCount

RangeCount()는 표현식 또는 필드에 있는 텍스트 및 숫자 값의 수를 반환합니다.

구문:

```
RangeCount(first_expr[, Expression])
```

반환 데이터 유형: 정수

인수:

이 함수의 인수에는 자체적으로 값 목록을 반환하는 인터 레코드 함수가 포함될 수 있습니다.

인수

인수	설명
first_expr	계산할 데이터가 포함된 표현식 또는 필드입니다.
Expression	계산할 데이터 범위가 포함된 옵션 표현식 또는 필드입니다.

제한 사항:

NULL 값은 계산되지 않습니다.

예 및 결과:

함수 예

예	결과
`RangeCount (1,2,4)`	3을 반환합니다.
`RangeCount (2,'xyz')`	2를 반환합니다.
`RangeCount (null( ))`	0을 반환합니다.
`RangeCount (2,'xyz', null())`	2를 반환합니다.

예제 스크립트를 앱에 추가하고 실행합니다. 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.

```
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
];
```

결과 테이블에 테이블 내 각 레코드에 대한 MyRangeCount의 반환된 값이 표시됩니다.

결과 테이블

RangeID	MyRangeCount
1	3
2	3
3	3
4	3
5	3
6	3

표현식이 포함된 예:

```
RangeCount (Above(MyField,1,3))
```

MyField의 세 결과에 포함된 값의 수를 반환합니다. **Above()** 함수의 첫 번째 인수를 1로 지정하고 두 번째 인수를 3으로 지정하면 충분한 행이 있을 경우 현재 행 위의 처음 세 필드에서 값을 반환하며 이는 **RangeCount()** 함수에 대한 입력으로 사용됩니다.

예에서 사용된 데이터:

샘플 데이터

MyField	RangeCount(Above(MyField,1,3))
10	0
2	1
8	2
18	3
5	3
9	3

예에서 사용된 데이터:

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
```

```
18
5
9
] ;
```

관련 항목:

Count - 차트 함수 (page 342)

RangeFractile

RangeFractile()은 숫자 범위의 n번째 **fractile**(사분위수)에 해당하는 값을 반환합니다.

RangeFractile()은 분위수를 계산할 때 가장 가까운 순위 사이의 선형 보간을 사용합니다.

구문:

```
RangeFractile(fractile, first_expr[, Expression])
```

반환 데이터 유형: 숫자

인수:

이 함수의 인수에는 자체적으로 값 목록을 반환하는 인터 레코드 함수가 포함될 수 있습니다.

인수

인수	설명
fractile	분위수에 대응하는 0 ~ 1 사이의 숫자(분위수로 표현된 사분위수)가 계산됩니다.
first_expr	측정할 데이터가 포함된 표현식 또는 필드입니다.
Expression	측정할 데이터 범위가 포함된 옵션 표현식 또는 필드입니다.

예 및 결과:

함수 예

예	결과
`RangeFractile (0.24,1,2,4,6)`	1.72를 반환합니다.
`RangeFractile(0.5,1,2,3,4,6)`	3을 반환합니다.
`RangeFractile (0.5,1,2,5,6)`	3.5를 반환합니다.

예제 스크립트를 앱에 추가하고 실행합니다. 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.

```
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
];
```

결과 테이블에 테이블 내 각 레코드에 대한 MyRangeFrac의 반환된 값이 표시됩니다.

결과 테이블

RangeID	MyRangeFrac
1	6
2	3
3	8
4	11
5	5
6	4

표현식이 포함된 예:

```
RangeFractile (0.5, Above(Sum(MyField),0,3))
```

이 예에서는 인터 레코드 함수 **Above()**에 옵션 offset 및 count 인수가 포함되었습니다. 이는 모든 범위 함수에 대한 입력으로 사용할 수 있는 다양한 결과를 산출합니다. 이 예에서 `Above(Sum(MyField),0,3)`는 현재 행과 그 위의 두 행에 대한 `MyField` 값을 반환합니다. 이 값은 **RangeFractile()** 함수에 대한 입력을 제공합니다. 따라서 아래 테이블의 맨 아래 행의 경우 이것은 `RangeFractile(0.5, 3,4,6)`와 동일합니다. 즉, 수열 3, 4 및 6에 대한 0.5 분위수를 계산합니다. 아래 테이블의 처음 두 행에서는 해당 범위의 값 수가 적절히 줄어들며 현재 행 위에는 행이 없습니다. 다른 인터 레코드 함수에서도 유사한 결과가 나옵니다.

샘플 데이터

MyField	RangeFractile(0.5, Above(Sum(MyField),0,3))
1	1
2	1.5
3	2
4	3
5	4
6	5

예에서 사용된 데이터:

```
RangeTab:
LOAD * INLINE [
```

```
MyField
1
2
3
4
5
6
] ;
```

관련 항목:

- *Above - 차트 함수 (page 1228)*
- *Fractile - 차트 함수 (page 395)*

RangeIRR

RangeIRR()은 입력 값으로 표현된 일련의 현금 흐름에 대한 내부 수익률을 반환합니다.

내부 수익률은 정기적인 기간에 일어나는 납입(음수) 및 수입(양수)으로 구성된 투자에 대해 수급되는 이자율입니다.

이 함수는 내부 반환율(IRR)을 계산하기 위해 간소화된 버전의 Newton 방법을 사용합니다.

구문:

```
RangeIRR(value[, value][, Expression])
```

반환 데이터 유형: 숫자

인수

인수	설명
value	선택적인 세 번째 파라메타가 포함된 인터 레코드 함수에서 반환된 단일 값 또는 값의 범위입니다. 이 함수를 계산하려면 최소 하나 이상의 양수와 하나 이상의 음수 값이 필요합니다.
Expression	측정할 데이터 범위가 포함된 옵션 표현식 또는 필드입니다.

제한 사항:

텍스트 값, NULL 값, 누락된 값은 무시됩니다.

예 테이블

예	결과
`RangeIRR(-70000,12000,15000,18000,21000,26000)`	0.0866을 반환합니다.

예	결과

예제 스크립트를 앱에 추가하고 실행합니다. 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.
`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 '|');`
	결과 테이블에 테이블 내 각 레코드에 대한 RangeIRR의 반환된 값이 표시됩니다.
	RangeID RangeIRR
	1 0.0639
	2 0.8708
	3 -
	4 5.8419
	5 0.9318
	6 -0.2566

관련 항목:

- *인터 레코드 함수 (page 1225)*

RangeKurtosis

RangeKurtosis()는 숫자 범위의 첨도에 해당하는 값을 반환합니다.

구문:

```
RangeKurtosis(first_expr[, Expression])
```

반환 데이터 유형: 숫자

인수:

이 함수의 인수에는 자체적으로 값 목록을 반환하는 인터 레코드 함수가 포함될 수 있습니다.

인수

인수	설명
first_expr	측정할 데이터가 포함된 표현식 또는 필드입니다.
Expression	측정할 데이터 범위가 포함된 옵션 표현식 또는 필드입니다.

제한 사항:

숫자 값이 발견되지 않으면 NULL이 반환됩니다.

예 및 결과:

함수 예

예	결과
`RangeKurtosis (1,2,4,7)`	-0.28571428571429를 반환합니다.

관련 항목:

- *Kurtosis - 차트 함수 (page 402)*

RangeMax

RangeMax()는 표현식 또는 필드 내에서 찾은 가장 높은 숫자 값을 반환합니다.

구문:

```
RangeMax(first_expr[, Expression])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
first_expr	측정할 데이터가 포함된 표현식 또는 필드입니다.
Expression	측정할 데이터 범위가 포함된 옵션 표현식 또는 필드입니다.

제한 사항:

숫자 값이 발견되지 않으면 NULL이 반환됩니다.

예 및 결과:

함수 예

예	결과
RangeMax (1,2,4)	4를 반환합니다.
RangeMax (1,'xyz')	1을 반환합니다.
RangeMax (null( ), 'abc')	NULL를 반환합니다.

예제 스크립트를 앱에 추가하고 실행합니다. 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.

```
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
];
```

결과 테이블에 테이블 내 각 레코드에 대한 MyRangeMax의 반환된 값이 표시됩니다.

결과 테이블

RangeID	MyRangeMax
1	10
2	7
3	8
4	18
5	9
6	9

표현식이 포함된 예:

```
RangeMax (Above(MyField,0,3))
```

현재 행과 현재 행 위의 두 행에서 계산된 **MyField**의 세 값의 범위 중 최대값을 반환합니다. 세 번째 인수를 3으로 지정하면, 위에 충분한 행이 있을 경우 **Above()** 함수에서 세 값을 반환하며, 이는 **RangeMax()** 함수에 대한 입력으로 사용됩니다.

예에서 사용된 데이터:

*예상대로 예제가 작동하도록 **MyField**의 정렬을 비활성화합니다.*

샘플 데이터

MyField	RangeMax (Above(Sum(MyField),1,3))
10	10
2	10
8	10
18	18
5	18
9	18

예에서 사용된 데이터:

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
] ;
```

RangeMaxString

RangeMaxString()은 표현식 또는 필드에서 찾은 텍스트 정렬 순서의 마지막 값을 반환합니다.

구문:

```
RangeMaxString(first_expr[, Expression])
```

반환 데이터 유형: 문자열

인수:

이 함수의 인수에는 자체적으로 값 목록을 반환하는 인터 레코드 함수가 포함될 수 있습니다.

인수

인수	설명
first_expr	측정할 데이터가 포함된 표현식 또는 필드입니다.
Expression	측정할 데이터 범위가 포함된 옵션 표현식 또는 필드입니다.

예 및 결과:

함수 예

예	결과
`RangeMaxString (1,2,4)`	4를 반환합니다.
`RangeMaxString ('xyz','abc')`	'xyz'를 반환합니다.
`RangeMaxString (5,'abc')`	'abc'를 반환합니다.
`RangeMaxString (null( ))`	NULL를 반환합니다.

표현식이 포함된 예:

```
RangeMaxString (Above(MaxString(MyField),0,3))
```

현재 행과 현재 행 위의 두 행에서 평가된 **MaxString(MyField)** 함수의 세 결과 중 마지막 값(텍스트 정렬 순서에서)을 반환합니다.

예에서 사용된 데이터:

*예상대로 예제가 작동하도록 **MyField**의 정렬을 비활성화합니다.*

샘플 데이터

MyField	RangeMaxString(Above(MaxString(MyField),0,3))
10	10
abc	abc
8	abc
def	def
xyz	xyz
9	xyz

예에서 사용된 데이터:

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
8
'def'
'xyz'
9
] ;
```

관련 항목:

MaxString - 차트 함수 (page 516)

RangeMin

RangeMin()은 표현식 또는 필드 내에서 찾은 가장 낮은 숫자 값을 반환합니다.

구문:

```
RangeMin(first_expr[, Expression])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
first_expr	측정할 데이터가 포함된 표현식 또는 필드입니다.
Expression	측정할 데이터 범위가 포함된 옵션 표현식 또는 필드입니다.

제한 사항:

숫자 값이 발견되지 않으면 NULL이 반환됩니다.

예 및 결과:

함수 예

예	결과
`RangeMin (1,2,4)`	1을 반환합니다.
`RangeMin (1,'xyz')`	1을 반환합니다.
`RangeMin (null( ), 'abc')`	NULL를 반환합니다.

예제 스크립트를 앱에 추가하고 실행합니다. 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.

```
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
];
```

결과 테이블에 테이블 내 각 레코드에 대한 MyRangeMin의 반환된 값이 표시됩니다.

결과 테이블

RangeID	MyRangeMin
1	5
2	2
3	2
4	9
5	5
6	2

표현식이 포함된 예:

```
RangeMin (Above(MyField,0,3)
```

현재 행과 현재 행 위의 두 행에서 계산된 **MyField**의 세 값의 범위 중 최소값을 반환합니다. 세 번째 인수를 3으로 지정하면, 위에 충분한 행이 있을 경우 **Above()** 함수에서 세 값을 반환하며, 이는 **RangeMin()** 함수에 대한 입력으로 사용됩니다.

예에서 사용된 데이터:

샘플 데이터

MyField	RangeMin(Above(MyField,0,3))
10	10
2	2
8	2
18	2
5	5
9	5

예에서 사용된 데이터:

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
] ;
```

관련 항목:

Min - 차트 함수 (page 329)

RangeMinString

RangeMinString()은 표현식 또는 필드에서 찾은 텍스트 정렬 순서의 첫 번째 값을 반환합니다.

구문:

```
RangeMinString(first_expr[, Expression])
```

반환 데이터 유형: 문자열

인수:

이 함수의 인수에는 자체적으로 값 목록을 반환하는 인터 레코드 함수가 포함될 수 있습니다.

인수

인수	설명
first_expr	측정할 데이터가 포함된 표현식 또는 필드입니다.
Expression	측정할 데이터 범위가 포함된 옵션 표현식 또는 필드입니다.

예 및 결과:

함수 예

예	결과
`RangeMinString (1,2,4)`	1을 반환합니다.
`RangeMinString ('xyz','abc')`	'abc'를 반환합니다.
`RangeMinString (5,'abc')`	5를 반환합니다.
`RangeMinString (null( ))`	NULL를 반환합니다.

표현식이 포함된 예:

```
RangeMinString (Above(MinString(MyField),0,3))
```

현재 행과 현재 행 위의 두 행에서 평가된 **MinString(MyField)** 함수의 세 결과 중 첫 번째 값(텍스트 정렬 순서에서)을 반환합니다.

예에서 사용된 데이터:

*예상대로 예제가 작동하도록 **MyField**의 정렬을 비활성화합니다.*

샘플 데이터

MyField	RangeMinString(Above(MinString(MyField),0,3))
10	10
abc	10
8	8
def	8
xyz	8
9	9

예에서 사용된 데이터:

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
8
'def'
'xyz'
9
] ;
```

관련 항목:

MinString - 차트 함수 (page 519)

RangeMissingCount

RangeMissingCount()는 표현식 또는 필드에 있는 숫자 이외의 값(NULL 포함)의 수를 반환합니다.

구문:

```
RangeMissingCount(first_expr[, Expression])
```

반환 데이터 유형: 정수

인수:

이 함수의 인수에는 자체적으로 값 목록을 반환하는 인터 레코드 함수가 포함될 수 있습니다.

인수

인수	설명
first_expr	계산할 데이터가 포함된 표현식 또는 필드입니다.
Expression	계산할 데이터 범위가 포함된 옵션 표현식 또는 필드입니다.

예 및 결과:

함수 예

예	결과
`RangeMissingCount (1,2,4)`	0을 반환합니다.
`RangeMissingCount (5,'abc')`	1을 반환합니다.
`RangeMissingCount (null( ))`	1을 반환합니다.

표현식이 포함된 예:

```
RangeMissingCount (Above(MinString(MyField),0,3))
```

현재 행과 현재 행 위의 두 행에서 평가된 **MinString(MyField)** 함수의 세 결과 중 숫자가 아닌 값의 수를 반환합니다.

*예상대로 예제가 작동하도록 **MyField**의 정렬을 비활성화합니다.*

샘플 데이터

MyField	RangeMissingCount (Above(MinString (MyField),0,3))	Explanation
10	2	이 행 위에 행이 없어 3개의 값 중 2개가 누락되었으므로 2를 반환합니다.
abc	2	현재 행 위에 1개의 행만 있으며 현재 행이 숫자가 아니므로('abc') 2를 반환합니다.
8	1	3개의 행 중 1개의 행에 숫자가 아닌 값('abc')이 포함되었으므로 1을 반환합니다.
def	2	3개의 행 중 2개의 행에 숫자가 아닌 값('def' 및 'abc')이 포함되었으므로 2를 반환합니다.
xyz	2	3개의 행 중 2개의 행에 숫자가 아닌 값(' xyz' 및 'def')이 포함되었으므로 2를 반환합니다.
9	2	3개의 행 중 2개의 행에 숫자가 아닌 값(' xyz' 및 'def')이 포함되었으므로 2를 반환합니다.

예에서 사용된 데이터:

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
8
'def'
'xyz'
```

```
9
] ;
```

관련 항목:

- *MissingCount - 차트 함수 (page 345)*

RangeMode

RangeMode()는 표현식 또는 필드에서 가장 발생 빈도가 높은 값(모드 값)을 찾습니다.

구문:

RangeMode(first_expr {, Expression}**)**

반환 데이터 유형: 숫자

인수:

이 함수의 인수에는 자체적으로 값 목록을 반환하는 인터 레코드 함수가 포함될 수 있습니다.

인수

인수	설명
first_expr	측정할 데이터가 포함된 표현식 또는 필드입니다.
Expression	측정할 데이터 범위가 포함된 옵션 표현식 또는 필드입니다.

제한 사항:

2개 이상의 값이 동등하게 가장 높은 빈도일 경우 NULL이 반환됩니다.

예 및 결과:

함수 예

예	결과
RangeMode (1,2,9,2,4)	2를 반환합니다.
RangeMode ('a',4,'a',4)	NULL를 반환합니다.
RangeMode (null( ))	NULL를 반환합니다.

예제 스크립트를 앱에 추가하고 실행합니다. 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.

```
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
];
```

결과 테이블에 테이블 내 각 레코드에 대한 **MyRangeMode**의 반환된 값이 표시됩니다.

결과 테이블

RangeID	MyRangMode
1	-
2	-
3	8
4	-
5	5
6	-

표현식이 포함된 예:

```
RangeMode (Above(MyField,0,3))
```

현재 행과 현재 행 위의 두 행에서 평가된 **MyField** 함수의 세 결과 중 가장 발생 빈도가 높은 값을 반환합니다. 세 번째 인수를 3으로 지정하면, 위에 충분한 행이 있을 경우 **Above()** 함수에서 세 값을 반환하며, 이는 **RangeMode()** 함수에 대한 입력으로 사용됩니다.

데이터 사용 예:

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
] ;
```

*예상대로 예제가 작동하도록 **MyField**의 정렬을 비활성화합니다.*

샘플 데이터

MyField	RangeMode(Above(MyField,0,3))
10	위에 행이 없어 단일 값이 가장 발생 빈도가 높으므로 10을 반환합니다.
2	-
8	-
18	-

MyField	RangeMode(Above(MyField,0,3))
5	-
9	-

관련 항목:

- *Mode - 차트 함수 (page 332)*

RangeNPV

RangeNPV()는 할인율 및 일련의 주기적인 미래 납입금(음수 값)과 수입(양수 값)을 기준으로 한 투자의 순 현재 가치를 반환합니다. 결과의 기본 숫자 서식은 **money**입니다.

주기적일 필요가 없는 현금 흐름인 경우, *RangeXNPV (page 1319)*를 참조하십시오.

구문:

```
RangeNPV(discount_rate, value[,value][, Expression])
```

반환 데이터 유형: 숫자

인수

인수	설명
discount_rate	기간별 이자율입니다.
value	각 기간 말에 일어나는 납입 또는 수입입니다. 각 값은 선택적인 세 번째 파라메타가 포함된 인터 레코드 함수에서 반환된 단일 값 또는 값의 범위일 수 있습니다.
Expression	측정할 데이터 범위가 포함된 옵션 표현식 또는 필드입니다.

제한 사항:

텍스트 값, NULL 값, 누락된 값은 무시됩니다.

예	결과
RangeNPV(0.1,-10000,3000,4200,6800)	1188.44를 반환합니다.

예	결과

예제 스크립트를 앱에 추가하고 실행합니다. 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.
`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 '|');`
	결과 테이블에 테이블 내 각 레코드에 대한 RangeNPV의 반환된 값이 표시됩니다.
	RangeID RangeNPV
	1 $-49.13
	2 $777.78
	3 $98.77
	4 $25.51
	5 $250.83
	6 $20.40

관련 항목:

- *인터 레코드 함수 (page 1225)*

RangeNullCount

RangeNullCount()는 표현식 또는 필드에서 NULL 값의 수를 찾습니다.

구문:

```
RangeNullCount(first_expr [, Expression])
```

반환 데이터 유형: 정수

인수:

이 함수의 인수에는 자체적으로 값 목록을 반환하는 인터 레코드 함수가 포함될 수 있습니다.

인수

인수	설명
first_expr	측정할 데이터가 포함된 표현식 또는 필드입니다.
Expression	측정할 데이터 범위가 포함된 옵션 표현식 또는 필드입니다.

예 및 결과:

함수 예

예	결과
`RangeNullCount (1,2,4)`	0을 반환합니다.
`RangeNullCount (5,'abc')`	0을 반환합니다.
`RangeNullCount (null( ), null( ))`	2를 반환합니다.

표현식이 포함된 예:

```
RangeNullCount (Above(Sum(MyField),0,3))
```

현재 행과 현재 행 위의 두 행에서 평가된 **Sum(MyField)** 함수의 세 결과 중 NULL 값의 수를 반환합니다.

*아래 예에서 **MyField**를 복사해도 NULL 값이 발생하지 않습니다.*

샘플 데이터

MyField	RangeNullCount(Above(Sum(MyField),0,3))
10	이 행 위에 행이 없어 3개의 값 중 2개가 누락되었으므로(=NULL) 2를 반환합니다.
'abc'	현재 행 위에 하나의 행만 있어 세 값 중 하나가 누락되었으므로(=NULL) 1을 반환합니다.
8	세 행 중 어느 것도 NULL 값이 아니므로 0을 반환합니다.

예에서 사용된 데이터:

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
8
] ;
```

관련 항목:

NullCount - 차트 함수 (page 348)

RangeNumericCount

RangeNumericCount()는 표현식 또는 필드에서 숫자 값의 수를 찾습니다.

구문:

```
RangeNumericCount(first_expr[, Expression])
```

반환 데이터 유형: 정수

인수:

이 함수의 인수에는 자체적으로 값 목록을 반환하는 인터 레코드 함수가 포함될 수 있습니다.

인수

인수	설명
first_expr	측정할 데이터가 포함된 표현식 또는 필드입니다.
Expression	측정할 데이터 범위가 포함된 옵션 표현식 또는 필드입니다.

예 및 결과:

함수 예

예	결과
`RangeNumericCount (1,2,4)`	3을 반환합니다.
`RangeNumericCount (5,'abc')`	1을 반환합니다.
`RangeNumericCount (null( ))`	0을 반환합니다.

표현식이 포함된 예:

```
RangeNumericCount (Above(MaxString(MyField),0,3))
```

현재 행과 현재 행 위의 두 행에서 평가된 **MaxString(MyField)** 함수의 세 결과 중 숫자 값의 수를 반환합니다.

*예상대로 예제가 작동하도록 **MyField**의 정렬을 비활성화합니다.*

샘플 데이터

MyField	RangeNumericCount(Above(MaxString(MyField),0,3))
10	1
abc	1
8	2
def	1
xyz	1
9	1

예에서 사용된 데이터:

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
8
def
xyz
9
] ;
```

관련 항목:

NumericCount - 차트 함수 (page 351)

RangeOnly

RangeOnly()는 표현식이 하나의 고유한 값으로 평가되는 경우 값을 반환하는 dual 함수입니다. 해당되지 않는 경우에는 **NULL**을 반환합니다.

구문:

```
RangeOnly(first_expr[, Expression])
```

반환 데이터 유형: dual

인수:

이 함수의 인수에는 자체적으로 값 목록을 반환하는 인터 레코드 함수가 포함될 수 있습니다.

인수	설명
first_expr	측정할 데이터가 포함된 표현식 또는 필드입니다.
Expression	측정할 데이터 범위가 포함된 옵션 표현식 또는 필드입니다.

예 및 결과:

예	결과
`RangeOnly (1,2,4)`	NULL를 반환합니다.
`RangeOnly (5,'abc')`	NULL를 반환합니다.
`RangeOnly (null( ), 'abc')`	'abc'를 반환합니다.
`RangeOnly(10,10,10)`	10을 반환합니다.

관련 항목:

Only - 차트 함수 (page 335)

RangeSkew

RangeSkew()는 숫자 범위의 왜곡도에 해당하는 값을 반환합니다.

구문:

```
RangeSkew(first_expr[, Expression])
```

반환 데이터 유형: 숫자

인수:

이 함수의 인수에는 자체적으로 값 목록을 반환하는 인터 레코드 함수가 포함될 수 있습니다.

인수

인수	설명
first_expr	측정할 데이터가 포함된 표현식 또는 필드입니다.
Expression	측정할 데이터 범위가 포함된 옵션 표현식 또는 필드입니다.

제한 사항:

숫자 값이 발견되지 않으면 NULL이 반환됩니다.

예 및 결과:

함수 예

예	결과
`rangeskew (1,2,4)`	0.93521952958283을 반환합니다.
`rangeskew (above (SalesValue,0,3))`	현재 행과 현재 행 위의 두 행에서 계산된 above() 함수에서 반환된 세 값 범위의 가변 왜곡도를 반환합니다.

데이터 사용 예:

샘플 데이터

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
] ;
```

관련 항목:

Skew - 차트 함수 (page 433)

RangeStdev

RangeStdev()는 숫자 범위의 표준 편차를 찾습니다.

구문:

```
RangeStdev(first_expr[, Expression])
```

반환 데이터 유형: 숫자

인수:

이 함수의 인수에는 자체적으로 값 목록을 반환하는 인터 레코드 함수가 포함될 수 있습니다.

인수

인수	설명
first_expr	측정할 데이터가 포함된 표현식 또는 필드입니다.
Expression	측정할 데이터 범위가 포함된 옵션 표현식 또는 필드입니다.

제한 사항:

숫자 값이 발견되지 않으면 NULL이 반환됩니다.

예 및 결과:

함수 예

예	결과
`RangeStdev (1,2,4)`	1.5275252316519를 반환합니다.
`RangeStdev (null( ))`	NULL를 반환합니다.

예	결과
RangeStdev (above (SalesValue),0,3))	현재 행과 현재 행 위의 두 행에서 계산된 above() 함수에서 반환된 세 값 범위의 가변 표준을 반환합니다.

데이터 사용 예:

샘플 데이터

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
] ;
```

관련 항목:

- *Stdev - 차트 함수 (page 435)*

RangeSum

RangeSum()은 값 범위의 합계를 반환합니다. 숫자가 아닌 모든 값은 0으로 처리됩니다.

구문:

```
RangeSum(first_expr[, Expression])
```

반환 데이터 유형: 숫자

인수:

이 함수의 인수에는 자체적으로 값 목록을 반환하는 인터 레코드 함수가 포함될 수 있습니다.

인수

인수	설명
first_expr	측정할 데이터가 포함된 표현식 또는 필드입니다.
Expression	측정할 데이터 범위가 포함된 옵션 표현식 또는 필드입니다.

제한 사항:

RangeSum 함수는 숫자가 아닌 모든 값을 0으로 취급합니다.

예 및 결과:

예

예	결과
`RangeSum (1,2,4)`	7을 반환합니다.
`RangeSum (5,'abc')`	5를 반환합니다.
`RangeSum (null( ))`	0을 반환합니다.

예제 스크립트를 앱에 추가하고 실행합니다. 결과를 보기 위해 결과 열에 나열된 필드를 앱의 시트에 추가합니다.

```
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
];
```

결과 테이블에 테이블 내 각 레코드에 대한 MyRangeSum의 반환된 값이 표시됩니다.

결과 테이블

RangeID	MyRangeSum
1	21
2	12
3	18
4	38
5	19
6	15

표현식이 포함된 예:

```
RangeSum (Above(MyField,0,3))
```

현재 행과 현재 행 위의 두 행에서 계산된 **MyField)**의 세 값의 합계를 반환합니다. 세 번째 인수를 3으로 지정하면, 위에 충분한 행이 있을 경우 **Above()** 함수에서 세 값을 반환하며, 이는 **RangeSum()** 함수에 대한 입력으로 사용됩니다.

예에서 사용된 데이터:

*예상대로 예제가 작동하도록 **MyField**의 정렬을 비활성화합니다.*

샘플 데이터

MyField	RangeSum(Above(MyField,0,3))
10	10
2	12
8	20
18	28
5	31
9	32

예에서 사용된 데이터:

```
RangeTab:
LOAD * INLINE [
MyField
10
2
8
18
5
9
] ;
```

관련 항목:

- *Sum - 차트 함수 (page 338)*
- *Above - 차트 함수 (page 1228)*

RangeTextCount

RangeTextCount()는 표현식 또는 필드에 있는 텍스트 값의 수를 반환합니다.

구문:

```
RangeTextCount(first_expr[, Expression])
```

반환 데이터 유형: 정수

인수:

이 함수의 인수에는 자체적으로 값 목록을 반환하는 인터 레코드 함수가 포함될 수 있습니다.

인수

인수	설명
first_expr	측정할 데이터가 포함된 표현식 또는 필드입니다.
Expression	측정할 데이터 범위가 포함된 옵션 표현식 또는 필드입니다.

예 및 결과:

함수 예

예	결과
`RangeTextCount (1,2,4)`	0을 반환합니다.
`RangeTextCount (5,'abc')`	1을 반환합니다.
`RangeTextCount (null( ))`	0을 반환합니다.

표현식이 포함된 예:

```
RangeTextCount (Above(MaxString(MyField),0,3))
```

현재 행과 현재 행 위의 두 행에서 평가된 **MaxString(MyField)** 함수의 세 결과 중 텍스트 값의 수를 반환합니다.

예에서 사용된 데이터:

*예상대로 예제가 작동하도록 **MyField**의 정렬을 비활성화합니다.*

데이터 예

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

예에서 사용된 데이터:

```
RangeTab:
LOAD * INLINE [
MyField
10
'abc'
8
null()
'xyz'
9
] ;
```

관련 항목:

TextCount - 차트 함수 (page 354)

RangeXIRR

RangeXIRR()은 주기적일 필요가 없는 현금 흐름의 일정에 대한 내부 수익률(연간)을 반환합니다. 일련의 주기적인 현금 흐름에 대한 내부 수익률을 계산하려면 **RangeIRR** 함수를 사용하십시오.

Qlik의 XIRR 기능(**XIRR()** 및 **RangeXIRR()** 함수)은 다음 등식을 통해 `Rate` 값을 구하여 올바른 XIRR 값을 확인합니다.

```
XNPV(Rate, pmt, date) = 0
```

이 등식은 Newton 방법의 간소화된 버전을 사용하여 구합니다.

구문:

```
RangeXIRR(value, date{, value, date})
```

반환 데이터 유형: 숫자

인수

인수	설명
value	date의 납입 일정에 해당하는 현금 흐름 또는 일련의 현금 흐름입니다. 일련의 값에 최소 하나 이상의 양수와 하나 이상의 음수 값이 포함되어 있어야 합니다.
date	현금 흐름 지급에 해당하는 납입 날짜 또는 납입 예정 날짜입니다.

이 함수를 사용할 때 다음 제한 사항이 적용됩니다.

- 텍스트 값, NULL 값, 누락된 값은 무시됩니다.
- 모든 납입금은 1년 365일을 기준으로 할인됩니다.
- 이 함수는 하나 이상의 유효한 음성 지불과 하나 이상의 유효한 양성 지불(해당 유효 날짜 포함)이 필요합니다. 이러한 결제가 제공되지 않으면 NULL 값이 반환됩니다.

다음 항목은 이 함수를 사용하는 데 도움이 될 수 있습니다.

- *RangeXNPV (page 1319)*: 이 함수를 사용하여 주기적일 필요가 없는 현금 흐름 일정에 대한 순 현재 가치를 계산합니다.
- *XIRR (page 368)*: **XIRR()** 함수는 현금 흐름 일정(주기적일 필요가 없음)에 대해 집계된 내부 수익률(연간)을 계산합니다.

Qlik Sense 클라이언트 관리의 다양한 버전 간에는 이 함수에서 사용하는 기본 알고리즘에 변형이 있습니다. 알고리즘의 최근 업데이트에 대한 자세한 내용은 지원 문서 XIRR 함수 수정 및 업데이트를 참조하십시오.

예 및 결과:

예 및 결과

예	결과
`RangeXIRR(-2500,'2008-01-01',2750,'2008-09-01')`	0.1532를 반환합니다.

관련 항목:

- *RangeIRR (page 1293)*
- *RangeXNPV (page 1319)*
- *XIRR (page 368)*
- XIRR 함수 수정 및 업데이트

RangeXNPV

RangeXNPV()는 **pmt** 및 **date**로 지정된 표현식에서 숫자 쌍으로 표현된 현금 흐름 일정(반드시 주기적일 필요는 없음)의 순 현재 가치를 반환합니다. 모든 납입금은 1년 365일을 기준으로 할인됩니다.

구문:

```
RangeXNPV(discount_rate, value, date{, value, date})
```

반환 데이터 유형: 숫자

인수

인수	설명
discount_rate	**discount_rate**는 결제 금액을 할인해야 하는 연간 할인율입니다.
value	date의 납입 일정에 해당하는 현금 흐름 또는 일련의 현금 흐름입니다. 각 값은 선택적인 세 번째 매개 변수가 포함된 인터 레코드 함수에서 반환된 단일 값 또는 값의 범위일 수 있습니다. 일련의 값에 최소 하나 이상의 양수와 하나 이상의 음수 값이 포함되어 있어야 합니다.
date	현금 흐름 지급에 해당하는 납입 날짜 또는 납입 예정 날짜입니다.

이 함수를 사용할 때 다음 제한 사항이 적용됩니다.

- 텍스트 값, NULL 값, 누락된 값은 무시됩니다.
- 모든 납입금은 1년 365일을 기준으로 할인됩니다.

예 - 스크립트

로드 스크립트 및 결과

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- RangeTab3이라는 테이블에 포함된 재무 데이터.
- **RangeXNPV()** 함수를 사용하여 순 현재 가치를 계산합니다.

로드 스크립트

```
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 '|');
```

결과

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 필드를 차원으로 추가합니다.

- RangeID
- RangeXNPV

결과 테이블

RangeID	RangeXNPV
1	-$9.09
2	-$0.00
3	$13.64

예 - 차트 표현식

로드 스크립트 및 차트 표현식

개요

데이터 로드 편집기를 열고 아래의 로드 스크립트를 새 탭에 추가합니다.

로드 스크립트에는 다음이 포함됩니다.

- RangeTab3이라는 테이블에 포함된 재무 데이터.
- **RangeXNPV()** 함수를 사용하여 순 현재 가치를 계산합니다.

로드 스크립트

```
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 '|');
```

결과

다음과 같이 하십시오.

데이터를 로드하고 시트를 엽니다. 새 테이블을 만들고 다음 계산을 측정값으로 추가합니다.

```
=RangeXNPV(0.1, -2500,'2008-01-01',2750,'2008-09-01')
```

결과 테이블

=XIRR(Payments, Date)
$80.25

관련 항목:

XNPV (page 374)

8.22 관계형 함수

이미 집계된 숫자를 사용하여 차트에서 개별 차원 값의 속성을 계산하는 함수 그룹입니다.

함수 출력이 데이터 포인트 자체의 값뿐만 아니라 다른 데이터 포인트에 대한 값의 관계에 따라서도 다르다는 점에서 이 함수는 관계형입니다. 예를 들어 다른 차원 값과의 비교 없이 순위를 계산할 수 없습니다.

이 함수는 차트 표현식에서만 사용할 수 있습니다. 로드 스크립트에서 사용할 수 없습니다.

차원은 비교에 필요한 다른 데이터 포인트를 정의하기 때문에 차트에 필요합니다. 따라서 관계형 함수는 무차원 차트(예: KPI 개체)에서 의미가 없습니다.

순위 함수

이 함수를 사용하면 0 값 제거가 자동으로 비활성화됩니다. NULL 값은 무시됩니다.

Rank

Rank()는 표현식에서 차트의 행을 평가하고, 각 행에 대해 표현식에서 평가한 차원 값의 상대적인 위치를 표시합니다. 표현식을 평가하는 경우 이 함수는 결과를 현재 열 세그먼트가 포함된 다른 행의 결과와 비교한 다음 세그먼트 내 현재 행의 순위를 반환합니다.

```
Rank - 차트 함수([TOTAL [<fld {, fld}>]] expr[, mode[, fmt]])
```

HRank

HRank()는 표현식을 평가하고 결과를 피벗 테이블의 현재 행 세그먼트가 포함된 다른 열의 결과와 비교합니다. 그런 다음 이 함수는 세그먼트 내에서 현재 열의 순위를 반환합니다.

```
HRank - 차트 함수([TOTAL] expr[, mode[, fmt]])
```

클러스터링 함수

KMeans2D

속성 그룹 **사이트 라이센스**에는 Qlik Sense 시스템 라이센스와 관련된 속성이 포함되어 있습니다. 모든 필드는 필수이며 비워 둘 수 없습니다.

사이트 라이센스 속성

속성 이름	설명
소유자 이름	Qlik Sense 제품 소유자의 사용자 이름입니다.
소유자 조직	Qlik Sense 제품 소유자가 속한 조직의 이름입니다.
일련 번호	Qlik Sense 소프트웨어에 할당된 일련 번호입니다.
콘트롤 번호	Qlik Sense 소프트웨어에 할당된 콘트롤 번호입니다.
LEF 액세스	Qlik Sense 소프트웨어에 할당된 라이센스 활성화 파일(LEF)입니다.

KMeans2D()는 k-평균클러스터링을 적용하여 차트의 행을 평가하고 각 차트 행에 이 데이터 포인트가 할당된 클러스터의 클러스터 ID를 표시합니다. 클러스터링 알고리즘에서 사용되는 열은 매개 변수 coordinate_1 및 coordinate_2에 의해 각각 결정됩니다. 둘 다 집계 열입니다. 생성된 클러스터 수는 num_clusters 매개 변수에 의해 결정됩니다. 데이터는 선택적으로 표준 매개 변수로 정규화할 수 있습니다.

```
KMeans2D - 차트 함수(num_clusters, coordinate_1, coordinate_2 [, norm])
```

KMeansND

KMeansND()는 k-평균클러스터링을 적용하여 차트의 행을 평가하고 각 차트 행에 이 데이터 포인트가 할당된 클러스터의 클러스터 ID를 표시합니다. 클러스터링 알고리즘에서 사용되는 열은 매개 변수 coordinate_1, coordinate_2 등(최대 n열)에 의해 결정됩니다. 모두 집계 열입니다. 생성된 클러스터 수는 num_clusters 매개 변수에 의해 결정됩니다.

```
KMeansND - 차트 함수(num_clusters, num_iter, coordinate_1, coordinate_2
[,coordinate_3 [, ...]])
```

KMeansCentroid2D

KMeansCentroid2D()는 k-평균클러스터링을 적용하여 차트의 행을 평가하고 각 차트 행에 이 데이터 포인트가 할당된 클러스터의 원하는 좌표를 표시합니다. 클러스터링 알고리즘에서 사용되는 열은 매개 변수 coordinate_1 및 coordinate_2에 의해 각각 결정됩니다. 둘 다 집계 열입니다. 생성된 클러스터 수는 num_clusters 매개 변수에 의해 결정됩니다. 데이터는 선택적으로 표준 매개 변수로 정규화할 수 있습니다.

```
KMeansCentroid2D - 차트 함수(num_clusters, coordinate_no, coordinate_1,
coordinate_2 [, norm])
```

KMeansCentroidND

KMeansCentroidND()는 k-평균클러스터링을 적용하여 차트의 행을 평가하고 각 차트 행에 이 데이터 포인트가 할당된 클러스터의 원하는 좌표를 표시합니다. 클러스터링 알고리즘에서 사용되는 열은 매개 변수 coordinate_1, coordinate_2 등(최대 n열)에 의해 결정됩니다. 모두 집계 열입니다. 생성된 클러스터 수는 num_clusters 매개 변수에 의해 결정됩니다.

```
KMeansCentroidND - 차트 함수(num_clusters, num_iter, coordinate_no, coordinate_1, coordinate_2 [,coordinate_3 [, ...]])
```

시계열 분해 함수

STL_Trend

STL_Trend는 시계열 분해 함수입니다. **STL_Seasonal** 및 **STL_Residual**과 함께 이 함수는 시계열을 계절성, 추세 및 잔차 구성 요소로 분해하는 데 사용됩니다. STL 알고리즘 컨텍스트에서 시계열 분해는 입력 메트릭 및 기타 매개 변수가 주어지면 반복되는 계절성 패턴과 일반적인 추세를 식별하는 데 사용됩니다. **STL_Trend** 함수는 시계열 데이터에서 계절성 패턴이나 주기와 관계없이 일반적인 추세를 식별합니다.

```
STL_Trend - 차트 함수(target_measure, period_int [,seasonal_smoother [,trend_smoother]])
```

STL_Seasonal

STL_Seasonal은 시계열 분해 함수입니다. **STL_Trend** 및 **STL_Residual**과 함께 이 함수는 시계열을 계절성, 추세 및 잔차 구성 요소로 분해하는 데 사용됩니다. STL 알고리즘 컨텍스트에서 시계열 분해는 입력 메트릭 및 기타 매개 변수가 주어지면 반복되는 계절성 패턴과 일반적인 추세를 식별하는 데 사용됩니다. **STL_Seasonal** 함수는 시계열 내에서 계절성 패턴을 식별하여 데이터에 표시되는 일반적인 추세와 구분할 수 있습니다.

```
STL_Seasonal - 차트 함수(target_measure, period_int [,seasonal_smoother [,trend_smoother]])
```

STL_Residual

STL_Residual은 시계열 분해 함수입니다. **STL_Seasonal** 및 **STL_Trend**와 함께 이 함수는 시계열을 계절성, 추세 및 잔차 구성 요소로 분해하는 데 사용됩니다. STL 알고리즘 컨텍스트에서 시계열 분해는 입력 메트릭 및 기타 매개 변수가 주어지면 반복되는 계절성 패턴과 일반적인 추세를 식별하는 데 사용됩니다. 이 작업을 수행할 때 입력 메트릭의 변동 중 일부는 계절성 구성 요소나 추세 구성 요소에 맞지 않으며 잔여 구성 요소로 정의됩니다. **STL_Residual** 차트 함수는 계산의 이 부분을 캡처합니다.

```
STL_Residual - 차트 함수(target_measure, period_int [,seasonal_smoother [,trend_smoother]])
```

Rank - 차트 함수

Rank()는 표현식에서 차트의 행을 평가하고, 각 행에 대해 표현식에서 평가한 차원 값의 상대적인 위치를 표시합니다. 표현식을 평가하는 경우 이 함수는 결과를 현재 열 세그먼트가 포함된 다른 행의 결과와 비교한 다음 세그먼트 내 현재 행의 순위를 반환합니다.

열 세그먼트

	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

테이블 이외의 차트에서는 현재 열 세그먼트가 차트의 일반표 해당 부분에 나타나는 대로 정의됩니다.

구문:

```
Rank([TOTAL] expr[, mode[, fmt]])
```

반환 데이터 유형: dual

인수:

인수

인수	설명
expr	측정할 데이터가 포함된 표현식 또는 필드입니다.
mode	함수 결과의 숫자 표현을 지정합니다.
fmt	함수 결과의 텍스트 표현을 지정합니다.
TOTAL	차트가 1차원이거나 표현식 앞에 **TOTAL** 한정자가 오는 경우 이 함수는 전체 열에 따라 평가됩니다. 테이블 또는 테이블과 동등한 것에 여러 세로 차원이 있을 경우 현재 열 세그먼트에는 필드 간 정렬 순서에서 마지막 차원이 표시되는 열을 제외하고 모든 차원 열 내의 현재 행과 같은 값을 가진 행만 포함됩니다.

순위는 이중 값으로 반환되며, 각 행은 1과 현재 열 세그먼트의 행 수 사이의 정수인 고유 순위를 갖습니다.

여러 행이 동일한 순위를 공유할 경우 **mode** 및 **fmt** 매개 변수를 사용하여 텍스트 및 숫자 표현을 제어할 수 있습니다.

mode
두 번째 인수인 **mode**에는 다음 값을 적용할 수 있습니다.

mode 예

값	설명
0(기본값)	공유 그룹 내의 모든 순위가 전체 순위의 중간 값에서 낮은 쪽에 속할 경우 공유 그룹 내의 모든 행에 최저 순위가 부여됩니다. 공유 그룹 내의 모든 순위가 전체 순위의 중간 값에서 높은 쪽에 속할 경우 공유 그룹 내의 모든 행에 최고 순위가 부여됩니다. 공유 그룹 내의 순위가 전체 순위의 중간 값에 분포할 경우 모든 행에 전체 열 세그먼트의 최고 및 최저 순위의 평균에 해당하는 값이 부여됩니다.
1	모든 행에 최저 순위를 부여합니다.
2	모든 행에 평균 순위를 부여합니다.
3	모든 행에 최고 순위를 부여합니다.
4	첫 번째 행에 최저 순위를 부여하고, 각 행마다 순위가 하나씩 올라갑니다.

fmt
세 번째 인수인 **fmt**에는 다음 값을 적용할 수 있습니다.

fmt예

값	설명
0(기본값)	모든 행에 낮은 값 - 높은 값을 부여합니다(예: 3 - 4).
1	모든 행에 낮은 값을 부여합니다.
2	첫 번째 행에 낮은 값을 부여하고 나머지 행은 공백으로 채웁니다.

mode 4 및 **fmt** 2에 해당하는 행의 순서는 차트 차원의 정렬 순서에 따라 결정됩니다.

예 및 결과:

차원 Product 및 Sales에서 두 시각화를 만들고, Product 및 UnitSales에서 또 다른 시각화를 만듭니다. 아래 테이블에 표시된 것처럼 측정값을 추가합니다.

순위 예

예	결과
예 1. 차원 `Customer` 및 `Sales`와 측정값 `Rank(Sales)`를 사용하여 테이블을 만듭니다.	결과는 차원의 정렬 순서에 따라 달라집니다. 테이블이 Customer로 정렬된 경우, 테이블에 Astrida에 대한 Sales 값에 이어 Betacab 순으로 해당하는 모든 값이 나열됩니다. Rank (Sales)의 결과는 Sales 값 12에 대해 10, Sales 값 13에 대해 9 등으로 표시되며, Sales 값 78의 경우 순위 값 1이 반환됩니다. 다음 열 세그먼트는 Betacab으로 시작되며, 여기서 세그먼트에 있는 Sales의 첫 번째 값은 12입니다. 이에 대한 Rank(Sales)의 순위 값은 11로 지정됩니다. 테이블이 Sales로 정렬된 경우, 열 세그먼트는 Sales 및 해당 Customer의 값으로 구성됩니다. 12에 해당하는 Sales 값이 두 개(Astrida 및 Betacab)이므로, 해당 열 세그먼트의 Rank(Sales) 값은 각 Customer에 대해 1-2입니다. 이는 Sales 값 12에 해당하는 Customer의 값이 둘이기 때문입니다. 4개의 값이 있다면 결과는 모든 행에서 1-4가 됩니다. 이는 인수 fmt의 기본값(0)에 대한 결과가 어떻게 나타날지 보여줍니다.
예 2. Customer 차원을 Product로 교체하고 측정값 `Rank(Sales,1,2)`를 추가합니다.	그러면 인수 **mode** 및 **fmt**가 각각 1과 2로 설정되므로, 각 열 세그먼트의 첫 번째 행에서 1이 반환되며 나머지 모든 행은 빈 채로 남습니다.

예 1의 결과 - Customer에서 테이블 정렬:

결과 테이블

Customer	Sales	Rank(Sales)
Astrida	12	10
Astrida	13	9

Customer	Sales	Rank(Sales)
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

예 1의 결과 - Sales에서 테이블 정렬:

결과 테이블

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

예에서 사용된 데이터:

```
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 '|');
```

관련 항목:

Sum - 차트 함수 (page 338)

HRank - 차트 함수

HRank()는 표현식을 평가하고 결과를 피벗 테이블의 현재 행 세그먼트가 포함된 다른 열의 결과와 비교합니다. 그런 다음 이 함수는 세그먼트 내에서 현재 열의 순위를 반환합니다.

구문:

```
HRank([ TOTAL ] expr [ , mode [, fmt ] ])
```

반환 데이터 유형: dual

이 함수는 피벗 테이블에서만 작동합니다. 다른 모든 차트 유형에서는 NULL을 반환합니다.

인수:

인수

인수	설명
expr	측정할 데이터가 포함된 표현식 또는 필드입니다.
mode	함수 결과의 숫자 표현을 지정합니다.
fmt	함수 결과의 텍스트 표현을 지정합니다.

인수	설명
TOTAL	차트가 1차원이거나 표현식 앞에 **TOTAL** 한정자가 오는 경우 이 함수는 전체 열에 따라 평가됩니다. 테이블 또는 테이블과 동등한 것에 여러 세로 차원이 있을 경우 현재 열 세그먼트에는 필드 간 정렬 순서에서 마지막 차원이 표시되는 열을 제외하고 모든 차원 열 내의 현재 행과 같은 값을 가진 행만 포함됩니다.

피벗 테이블이 1차원이거나 표현식 앞에**total** 한정자가 있을 경우 현재 행 세그먼트는 항상 전체 행과 동일합니다. 피벗 테이블에 여러 가로 차원이 있을 경우 현재 행 세그먼트에는 필드 간 정렬 순서에서 마지막 가로 차원이 표시되는 행을 제외하고 모든 차원 행 내의 현재 열과 같은 값을 가진 열만 포함됩니다.

순위는 이중 값으로 반환되며, 각 열은 1과 현재 행 세그먼트의 열 수 사이의 정수인 고유 순위를 갖습니다.

여러 열이 동일한 순위를 공유할 경우 **mode** 및 **format** 인수를 사용하여 텍스트 및 숫자 표현을 제어할 수 있습니다.

두 번째 인수 **mode**는 함수 결과의 숫자 표현을 지정합니다.

mode 예

값	설명
0(기본값)	공유 그룹 내의 모든 순위가 전체 순위의 중간 값에서 낮은 쪽에 속할 경우 공유 그룹 내의 모든 열에 최저 순위가 부여됩니다. 공유 그룹 내의 모든 순위가 전체 순위의 중간 값에서 높은 쪽에 속할 경우 공유 그룹 내의 모든 열에 최고 순위가 부여됩니다. 공유 그룹 내의 순위가 전체 순위의 중간 값에 분포할 경우 모든 행에 전체 열 세그먼트의 최고 및 최저 순위의 평균에 해당하는 값이 부여됩니다.
1	그룹 내 모든 열에 최저 순위를 부여합니다.
2	그룹 내 모든 열에 평균 순위를 부여합니다.
3	그룹 내 모든 열에 최고 순위를 부여합니다.
4	첫 번째 열에 최저 순위를 부여하고 그룹 내 각 열마다 순위가 하나씩 올라갑니다.

세 번째 인수 **format**은 함수 결과의 텍스트 표현을 지정합니다.

format 예

값	설명
0(기본값)	그룹 내 모든 열에 낮은 값**&' - '&**높은 값을 부여합니다(예: 3 - 4).
1	그룹 내 모든 열에 낮은 값을 부여합니다.
2	첫 번째 열에 낮은 값을 부여하고 그룹 내의 나머지 열은 공백으로 채웁니다.

mode 4 및 **format** 2에 해당하는 열의 순서는 차트 차원의 정렬 순서에 따라 결정됩니다.

```
HRank( sum( Sales ))

HRank( sum( Sales ), 2 )

HRank( sum( Sales ), 0, 1 )
```

K-평균으로 최적화: 실제 예

다음 예는 KMeans 클러스터링 및 Centroid 함수가 데이터 집합에 적용되는 실제 사용 사례를 보여 줍니다. KMeans 함수는 데이터 포인트를 유사성을 공유하는 클러스터로 분리합니다. KMeans 알고리즘이 구성 가능한 반복 횟수에 적용됨에 따라 클러스터가 더욱 간결해지고 차별화됩니다.

KMeans는 다양한 사용 사례에서 여러 분야에서 사용됩니다. 클러스터링 사용 사례의 몇 가지 예로는 고객 분류, 사기 탐지, 계정 손실 예측, 클라이언트 인센티브 대상 지정, 사이버 범죄 식별 및 배달 경로 최적화가 있습니다. KMeans 클러스터링 알고리즘은 기업이 패턴을 추론하고 서비스 제공을 최적화하려는 곳에서 점점 더 많이 사용되고 있습니다.

Qlik Sense KMeans 및 Centroid 함수

Qlik Sense는 유사성을 기반으로 데이터 포인트를 클러스터로 그룹화하는 두 개의 KMeans 함수를 제공합니다. *KMeans2D - 차트 함수 (page 1337)* 및 *KMeansND - 차트 함수 (page 1352)*를 참조하십시오. **KMeans2D** 함수는 두 개의 차원을 허용하며 **스캐터 차트**를 통해 결과를 시각화하는 데 적합합니다. **KMeansND** 함수는 세 개 이상의 차원을 허용합니다. 표준 차트에서 2D 결과를 개념화하기 쉽기 때문에 다음 데모에서는 두 개의 차원을 사용하여 **스캐터 차트**에 KMeans를 적용합니다. KMeans 클러스터링은 표현식별로 색을 지정하거나 이 예에 설명된 대로 차원별로 시각화할 수 있습니다.

Qlik Sense centroid 함수는 클러스터에 있는 모든 데이터 포인트의 산술 평균 위치를 결정하고 해당 클러스터의 중심 포인트 또는 centroid를 식별합니다. centroid 함수는 각 차트 행(또는 레코드)에 대해 이 데이터 포인트가 할당된 클러스터의 좌표를 표시합니다. *KMeansCentroid2D - 차트 함수 (page 1367)* 및 *KMeansCentroidND - 차트 함수 (page 1368)*를 참조하십시오.

사용 사례 및 개요 예

다음 예에서는 시뮬레이션된 실제 시나리오를 단계별로 진행합니다. 미국 뉴욕 주의 한 섬유 회사는 배송비를 최소화하여 비용을 줄여야 합니다. 이를 위한 한 가지 방법은 배포자와 가까운 창고를 재배치하는 것입니다. 이 회사는 뉴욕 주 전역에 118개의 배포자를 두고 있습니다. 다음 데모에서는 운영 관리자가 KMeans 함수를 통해 배포자를 클러스터하여 5개의 지역으로 분할한 다음 centroid 함수를 사용하여 해당 클러스터의 중심에 있는 5개의 최적 창고 위치를 식별하는 방법을 시뮬레이션합니다. 목표는 5개의 중앙 창고 위치를 식별하는 데 사용할 수 있는 매핑 좌표를 찾아내는 것입니다.

데이터 집합

이 데이터 집합은 실제 위도 및 경도 좌표를 사용하여 뉴욕 주에서 임의로 생성된 이름과 주소를 기반으로 합니다. 데이터 집합에는 id, first_name, last_name, telephone, address, city, state, zip, latitude, longitude 등 10 개의 열이 포함됩니다. 데이터 집합은 로컬로 다운로드한 다음 Qlik Sense로 업로드하거나 데이터 로드 편집기에 인라인할 수 있는 파일로 아래에서 제공됩니다. 만들어지는 앱의 이름은 *Distributors KMeans and Centroid*으로 지정되며 앱의 첫 번째 시트의 이름은 *Distribution cluster analysis*로 지정됩니다.

샘플 데이터 파일을 다운로드하려면 다음 링크를 선택합니다. DistributorData.csv

Distributor 데이터 집합: Qlik Sense에서 데이터 로드 편집기에 대한 인라인 로드 (page 1335)

제목: DistributorData

총 레코드 수: 118개

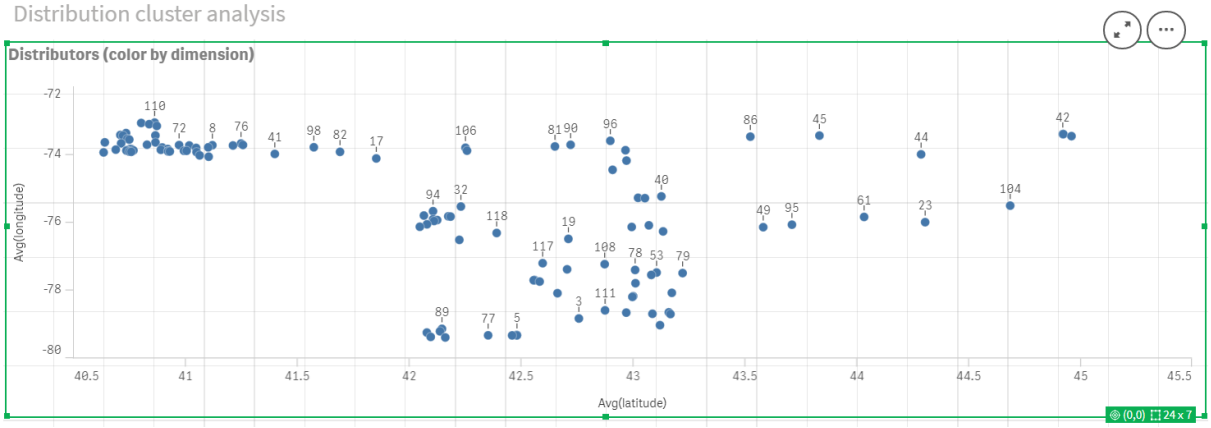
KMeans2D 함수 적용

이 예에서는 *DistributorData* 데이터 집합을 사용하여 **스캐터** 차트의 구성을 보여 주고 **KMeans2D** 함수가 적용되며 차트는 차원별로 색이 지정됩니다.

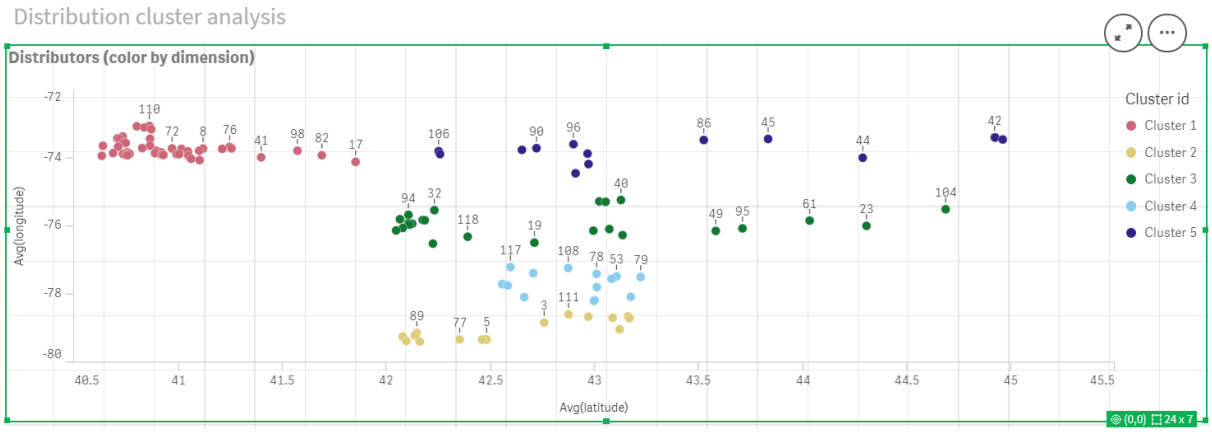
Qlik Sense KMeans 함수는 깊이 차(DeD)라는 방법을 사용하여 자동 클러스터링을 지원합니다. 사용자가 클러스터 수를 0으로 설정한 경우 해당 데이터 집합에 대한 최적의 클러스터 수가 결정됩니다. 그러나 이 예에서는 **num_clusters** 인수에 대한 변수가 만들어집니다(구문은 *KMeans2D - 차트 함수 (page 1337)* 참조). 따라서 원하는 클러스터 수(k = 5)는 변수로 지정됩니다.

1. **스캐터 차트**를 시트로 끌어와 *Distributors(by dimension)*으로 이름을 지정합니다.
2. 클러스터 수를 지정할 **변수**가 만들어집니다. **변수** 이름을 *vDistClusters*로 지정합니다. 변수 **정의**에 *5*를 입력합니다.
3. 차트에 대한 **데이터** 구성:
    a. **차원**에서 **거품** 필드에 대해 *id*를 선택합니다. **레이블**에 대해 *Cluster id*를 입력합니다.
    b. **측정값**에서 *Avg([latitude])*는 **X 축**에 대한 표현식입니다.
    c. **측정값**에서 *Avg([longitude])*는 **Y 축**에 대한 표현식입니다.
4. **모양**구성:
    a. **색 및 범례**에서 **색**에 대해 **사용자 지정**을 선택합니다.
    b. 차트 색을 지정하기 위해 **차원별**을 선택합니다.
    c. 다음 표현식을 입력합니다. *=pick(aggr(KMeans2D(vDistClusters,only(latitude),only(longitude)),id)+1, 'Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster 4', 'Cluster 5')*
    d. **영구 색**에 대한 확인란을 선택합니다.

차원별 KMeans 색이 지정되기 전에 스캐터 차트가 적용됨



차원별 KMeans 색이 지정된 후에 스캐터 차트가 적용됨



테이블 추가: *배포자*

관련 데이터에 빠르고 쉽게 액세스할 수 있는 테이블을 만드는 데 도움이 될 수 있습니다. **스캐터 차트**는 해당 배포자 이름이 있는 테이블이 참조용으로 추가되었지만 *ID*를 표시합니다.

1. **테이블**을 시트로 끌어와 *id*, *first_name* 및 *last_name* **열**(차원)을 추가하고 *Distributors*로 이름을 지정합니다.

테이블: 배포자 이름

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

막대형 차트 추가: *# observations per cluster*

창고 배포 시나리오의 경우 각 창고를 이용할 배포자 수를 알아야 도움이 됩니다. 따라서 각 클러스터에 할당된 배포자 수를 측정하는 **막대형 차트**를 만듭니다.

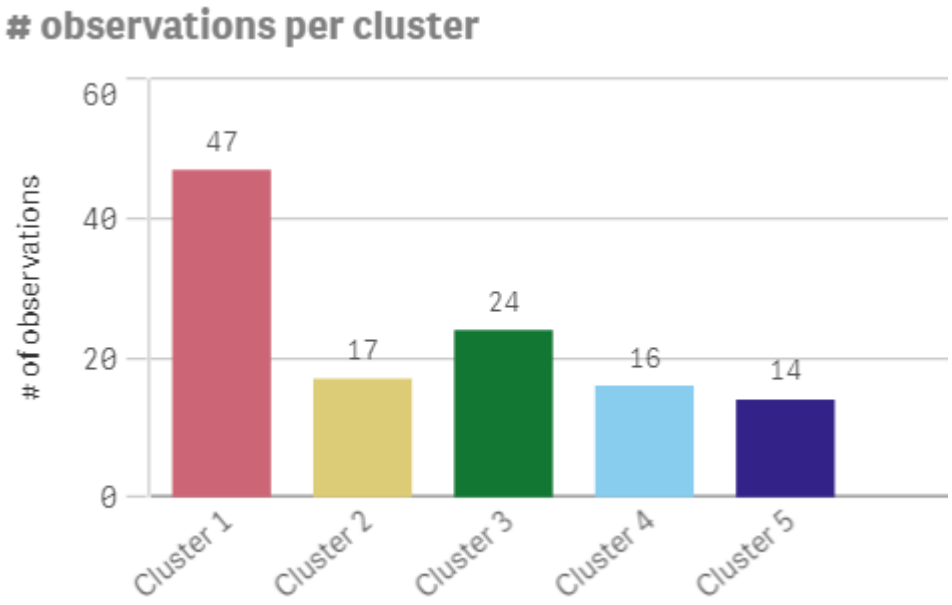
1. **막대형 차트**를 시트로 끌어서 놓습니다. 차트의 이름을 *# observations per cluster*로 지정합니다.
2. **막대형 차트**에 대한 **데이터** 구성:

a. *Clusters*라고 레이블이 지정된 **차원**을 추가합니다(표현식이 적용된 후 레이블을 추가할 수 있음). 다음 표현식을 입력합니다. *=pick(aggr(KMeans2D(vDistClusters,only(latitude),only(longitude)),id)+1, 'Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster 4', 'Cluster 5')*

b. *# of observations*라고 레이블이 지정된 **측정값**을 추가합니다. 다음 표현식을 입력합니다. *=count(aggr(KMeans2D(vDistClusters,only(latitude),only(longitude)),id))*

3. **모양**구성:
   a. **색 및 범례**에서 **색**에 대해 **사용자 지정**을 선택합니다.
   b. 차트 색을 지정하기 위해 **차원별**을 선택합니다.
   c. 다음 표현식을 입력합니다. *=pick(aggr(KMeans2D(vDistClusters,only(latitude),only(longitude)),id)+1, 'Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster 4', 'Cluster 5')*
   d. **영구 색**에 대한 확인란을 선택합니다.
   e. **범례 표시**를 끕니다.
   f. **프레젠테이션**에서 **값 레이블**을 **자동**으로 전환합니다.
   g. **X 축**에서: **클러스터**, **레이블만**을 선택합니다.

막대형 차트: # observations per cluster



Centroid2D 함수 적용

잠재적인 창고 위치의 좌표를 식별하는 **Centroid2D** 함수에 대한 두 번째 테이블을 추가합니다. 이 테이블에는 5개의 식별된 배포자 그룹의 중앙 위치(centroid 값)가 표시됩니다.

1. **테이블**을 시트로 끌어와 다음 열을 추가하고 *Cluster centroids*라는 이름을 지정합니다.
   a. *Clusters*라는 레이블이 지정된 **차원**을 추가합니다. 다음 표현식을 입력합니다.*=pick(aggr(KMeans2D(vDistClusters,only(latitude),only(longitude)),id)+1,'Warehouse 1','Warehouse 2','Warehouse 3','Warehouse 4','Warehouse 5'*)
   b. *latitude(D1)*라고 레이블이 지정된 **측정값**을 추가합니다. 다음 표현식을 입력합니다.*=only(aggr(KMeansCentroid2D(vDistClusters,0,only(latitude),only(longitude)),id))*
   매개 변수 **coordinate_no**는 첫 번째 차원(0)에 해당합니다. 이 경우 차원 *위도*는 x 축에 표시됩

니다. **CentroidND** 함수를 사용 중이고 최대 6개의 차원이 있는 경우 이러한 매개 변수 항목은 0, 1, 2, 3, 4, 5의 6개 값 중 하나일 수 있습니다.

c. *longitude(D2)*라고 레이블이 지정된 **측정값**을 추가합니다. 다음 표현식을 입력합니다.*=only (aggr(KMeansCentroid2D(vDistClusters,1,only(latitude),only(longitude)),id))*
이 표현식에서 매개 변수 **coordinate_no**는 두 번째 차원(1)에 해당합니다. 차원 *경도*는 y 축에 표시됩니다.

테이블: 클러스터 centroid 계산

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

Centroid 매핑

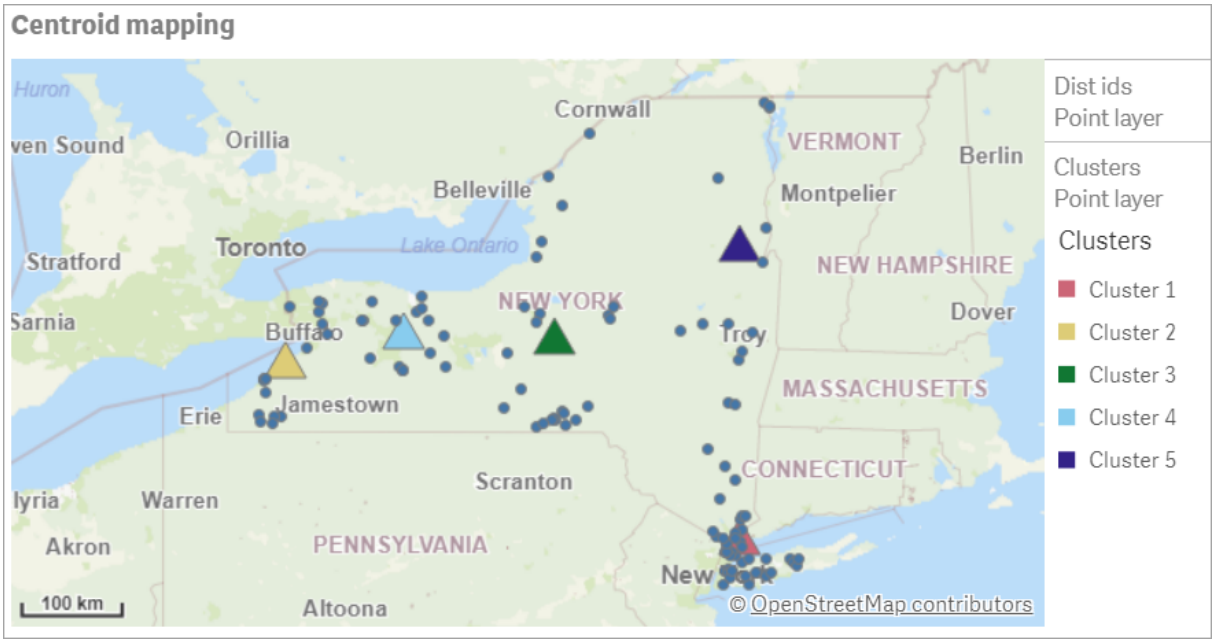
다음 단계에서 centroid를 매핑합니다. 시각화를 별도의 시트에 배치하는 것을 선호하는지는 앱 개발자에 따라 다릅니다.

1. *Centroid mapping*이라는 **맵**을 시트로 끌어서 놓습니다.
2. **레이어** 섹션에서. **레이어 추가**를 선택한 다음 **포인트 레이어**를 선택합니다.
    a. **필드** *ID*를 선택하고 *Dist ids* **레이블**를 추가합니다.
    b. **위치** 섹션에서 **Latitude 및 Longitude 필드** 확인란을 선택합니다.
    c. **위도**에서 *latitude* 필드를 선택합니다.
    d. **경도**에서 *longitude* 필드를 선택합니다.
    e. **크기 및 모양** 섹션에서 **모양**에 대해 **거품**을 선택하고 슬라이더에서 **크기**를 기본 설정으로 줄입니다.
    f. **색** 섹션에서 **단색**을 선택하고 **색**에서 파란색을 선택하고 **윤곽선** 색으로 회색을 선택합니다 (이러한 선택 사항도 기본 설정의 문제임).
3. **레이어** 섹션에서 **레이어 추가**를 선택한 다음 **포인트 레이어**를 선택하여 두 번째 **포인트 레이어**를 추가합니다.
    a. 다음 표현식을 입력합니다. *=aggr(KMeans2D(vDistClusters,only(latitude),only(longitude)),id)*
    b. **레이블**에서 *Clusters*를 추가합니다.
    c. **위치** 섹션에서 **Latitude 및 Longitude 필드** 확인란을 선택합니다.
    d. 이 경우 x 축을 따라 표시되는 **위도**의 경우 다음 표현식을 추가합니다. *=aggr (KMeansCentroid2D(vDistClusters,0,only(latitude),only(longitude)),id)*

e. 이 경우 y 축을 따라 표시되는 **경도**의 경우 다음 표현식을 추가합니다. *=aggr (KMeansCentroid2D(vDistClusters,1,only(latitude),only(longitude)),id)*
f. **크기 및 모양** 섹션에서 **모양**으로 **삼각형**을 선택하고 슬라이더에서 **크기**를 기본 설정으로 줄입니다.
g. **색 및 범례**에서 **색**에 대해 **사용자 지정**을 선택합니다.
h. 차트 색을 지정하기 위해 **차원별**을 선택합니다. 다음 표현식을 입력합니다. *=pick(aggr (KMeans2D(vDistClusters,only(latitude),only(longitude)),id)+1, 'Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster 4', 'Cluster 5')*
i. 차원에 *Clusters*라는 레이블을 지정합니다.

4. **맵 설정**에서 **투사**에 대해 **적응형**을 선택합니다. **측정 단위**로 **척도**를 선택합니다.

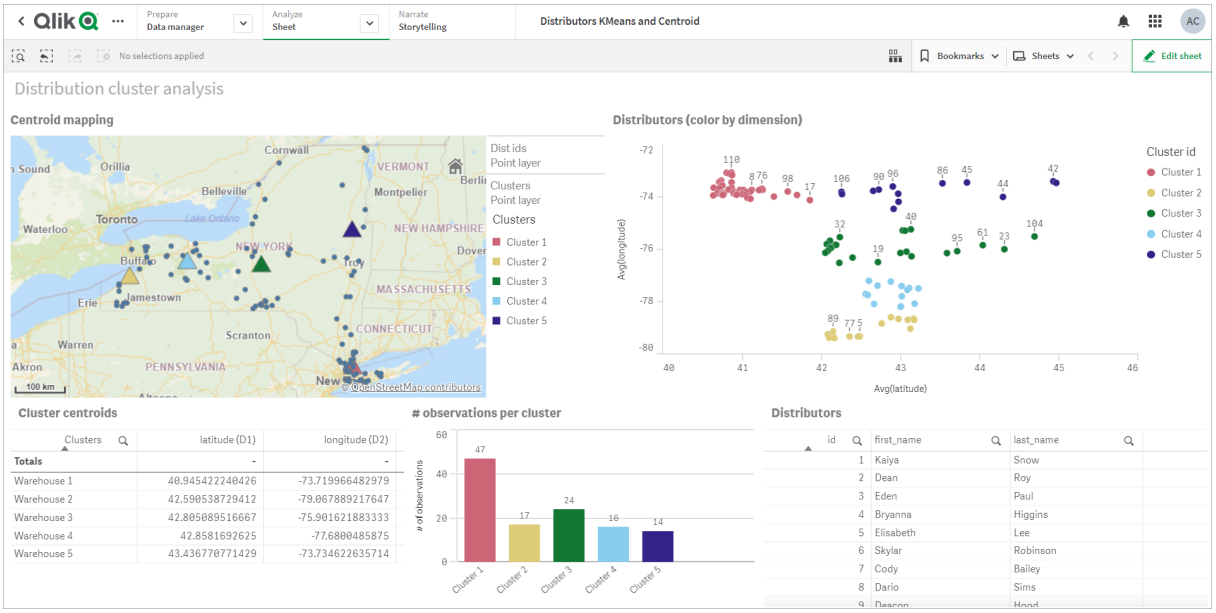
맵: 클러스터로 매핑된 centroid



결론

이 실제 시나리오에서 KMeans 함수를 사용하여 배포자는 유사성(이 경우 상호 근접성)을 기반으로 유사한 그룹 또는 클러스터로 분류되었습니다. Centroid 함수는 5개의 매핑 좌표를 식별하기 위해 이러한 클러스터에 적용되었습니다. 이러한 좌표는 창고를 구축하거나 찾을 수 있는 초기 중앙 위치를 제공합니다. centroid 함수는 **맵** 차트에 적용되어 앱 사용자가 주변 클러스터 데이터 포인트를 기준으로 centroid 위치를 시각화할 수 있습니다. 결과 좌표는 뉴욕 주에 있는 배포자의 배송 비용을 최소화할 수 있는 잠재적인 창고 위치를 나타냅니다.

앱: KMeans 및 centroid 분석 예



Distributor 데이터 집합: Qlik Sense에서 데이터 로드 편집기에 대한 인라인 로드

```
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 - 차트 함수

KMeans2D()는 k-평균클러스터링을 적용하여 차트의 행을 평가하고 각 차트 행에 이 데이터 포인트가 할당된 클러스터의 클러스터 ID를 표시합니다. 클러스터링 알고리즘에서 사용되는 열은 매개 변수 coordinate_1 및 coordinate_2에 의해 각각 결정됩니다. 둘 다 집계 열입니다. 생성된 클러스터 수는 num_clusters 매개 변수에 의해 결정됩니다. 데이터는 선택적으로 표준 매개 변수로 정규화할 수 있습니다.

KMeans2D는 데이터 포인트당 하나의 값을 반환합니다. 반환된 값은 이중 값이며 각 데이터 포인트가 할당된 클러스터에 해당하는 정수 값입니다.

구문:

```
KMeans2D(num_clusters, coordinate_1, coordinate_2 [, norm])
```

반환 데이터 유형: dual

인수:

인수

인수	설명
num_clusters	클러스터 수를 지정하는 정수입니다.
coordinate_1	첫 번째 좌표를 계산하는 집계는 일반적으로 차트로부터 만들 수 있는 스캐터 차트의 x 축입니다. 추가 매개 변수 coordinate_2는 두 번째 좌표를 계산합니다.
norm	선택적 정규화 방법이 KMeans 클러스터링 전에 데이터 집합에 적용됩니다. 가능한 값 : 정규화가 없는 경우 0 또는 '없음' z 점수 정규화의 경우 1 또는 'zscore' 최소-최대 정규화의 경우 2 또는 'minmax' 매개 변수가 제공되지 않거나 제공된 매개 변수가 잘못된 경우 정규화가 적용되지 않습니다. Z 점수는 기능 평균과 표준 편차를 기준으로 데이터를 정규화합니다. Z 점수는 각 기능이 동일한 척도를 갖도록 하지 않지만 이상값을 처리할 때 최소-최대보다 더 나은 접근 방식입니다. 최소-최대 정규화는 각각의 최솟값과 최댓값을 가져오고 각 데이터 포인트를 다시 계산하여 특성이 동일한 척도를 갖도록 합니다.

예: 차트 표현식
이 예제에서는 *Iris* 데이터 집합을 사용하여 스캐터 차트를 만든 다음 KMeans를 사용하여 표현식으로 데이터에 색을 지정합니다.

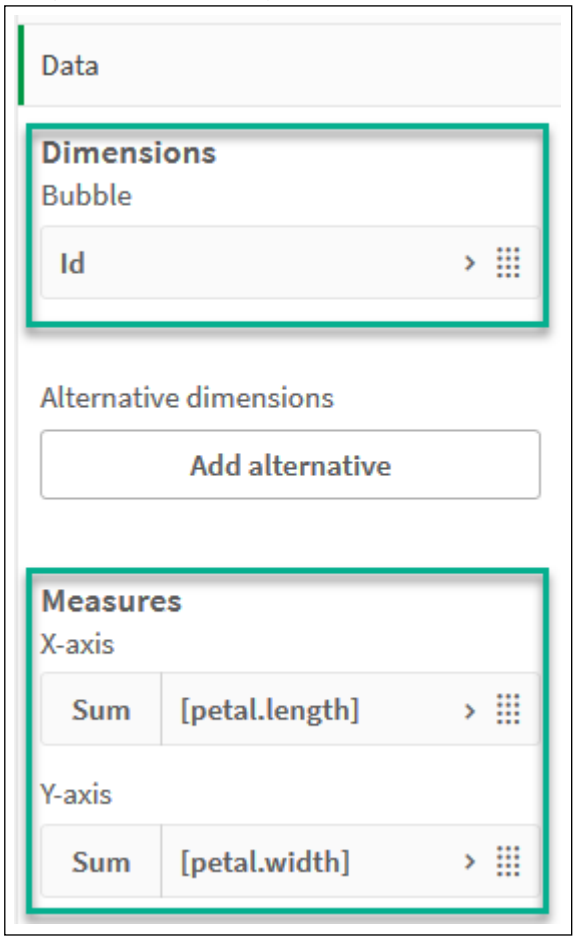
또한 *num_clusters* 인수에 대한 변수를 만든 다음 변수 입력 상자를 사용하여 클러스터 수를 변경합니다.

Iris 데이터 집합은 다양한 형식으로 공개적으로 사용할 수 있습니다. Qlik Sense의 데이터 로드 편집기를 사용하여 로드할 데이터를 인라인 테이블로 제공했습니다. 이 예에서는 데이터 테이블에 *Id* 열을 추가했습니다.

Qlik Sense에서 데이터 로드 후 다음을 수행합니다.

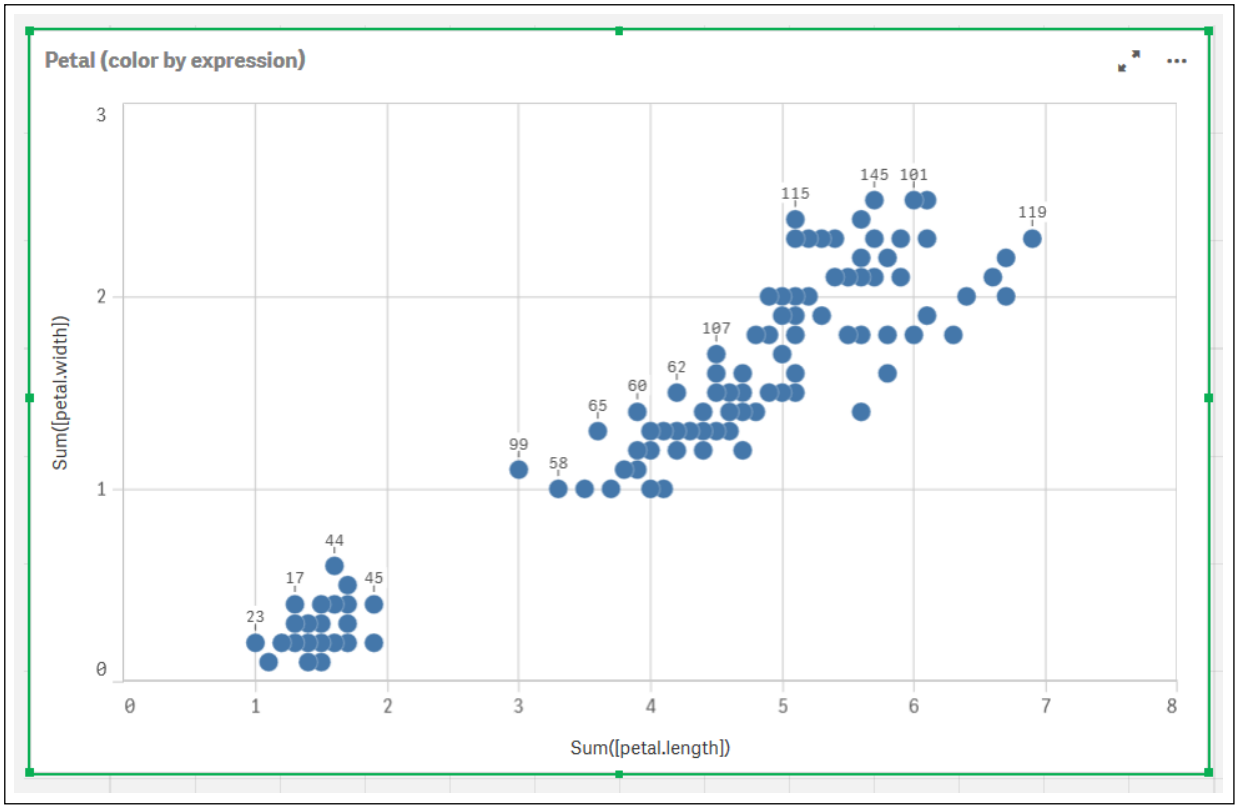
1. **스캐터 차트**를 새 시트로 끕니다. 차트의 이름을 *꽃잎(표현식으로 색 지정)*으로 지정합니다.
2. 변수를 만들어 클러스터 수를 지정합니다. 변수 **Name**에 *KmeansPetalClusters*를 입력합니다. 변수 **Definition**에 *=2*를 입력합니다.
3. 차트에 대한 **데이터**를 구성합니다.
    i. **차원**에서 **거품** 필드에 대해 *id*를 선택합니다. 레이블에 대해 클러스터 Id를 입력합니다.
    ii. **측정값**에서 **X 축** 표현식에 대해 *Sum([petal.length])*를 선택합니다.
    iii. **측정값**에서 **Y 축** 표현식에 대해 *Sum([petal.width])*를 선택합니다.

    꽃잎(표현식 기준 색 지정) 차트의 데이터 설정



데이터 포인트가 차트에 그려집니다.

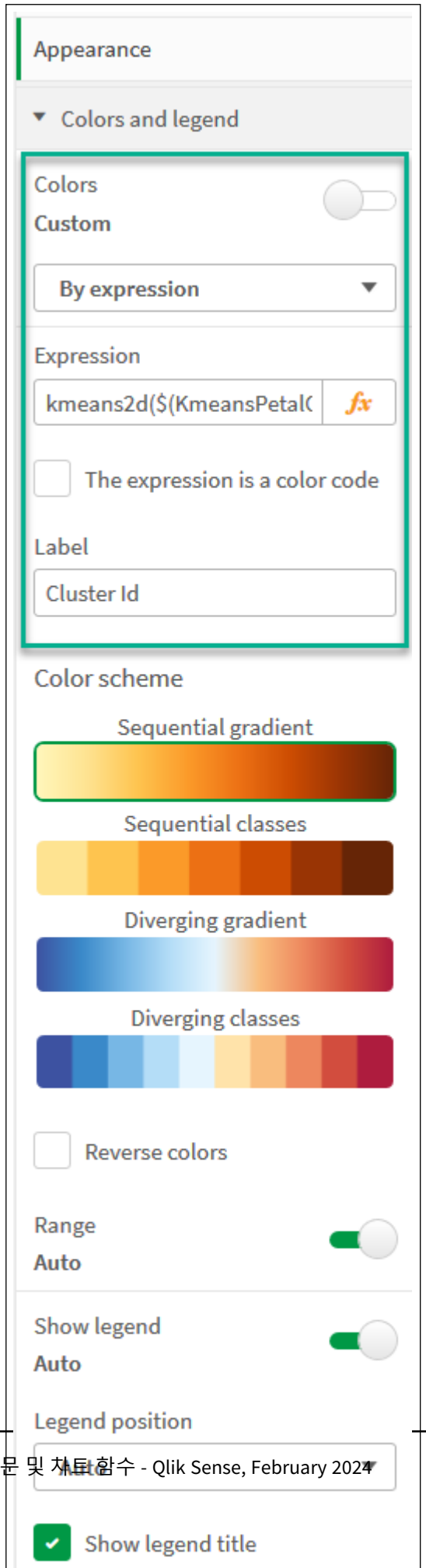
꽃잎(표현식 기준 색 지정) 차트의 데이터 포인트



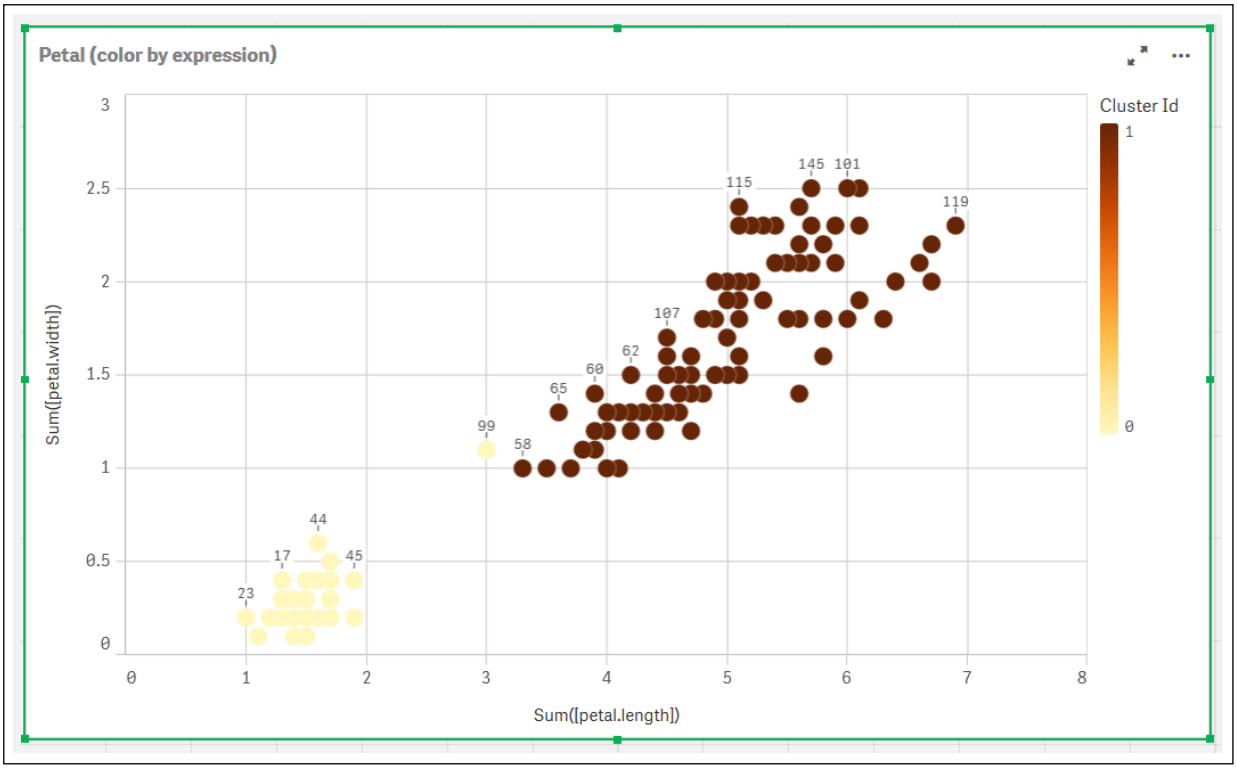
4. 차트의 **모양**을 구성합니다.
    i. **색 및 범례**에서 **색**에 대해 **사용자 지정**을 선택합니다.
    ii. **표현식 기준**으로 차트 색을 지정하도록 선택합니다.
    iii. **표현식**에 대해 다음을 입력합니다. *kmeans2d($(KmeansPetalClusters), Sum([petal.length]), Sum([petal.width]))*
    *KmeansPetalClusters*는 *2*로 설정한 변수입니다.
    또는 다음을 입력합니다. *kmeans2d(2, Sum([petal.length]), Sum([petal.width]))*
    iv. **표현식은 색상 코드입니다** 확인란을 선택 취소합니다.

v. **레이블**에 대해 다음을 입력합니다. *클러스터 Id*

꽃잎(표현식 기준 색 지정) 차트의 모양 설정


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

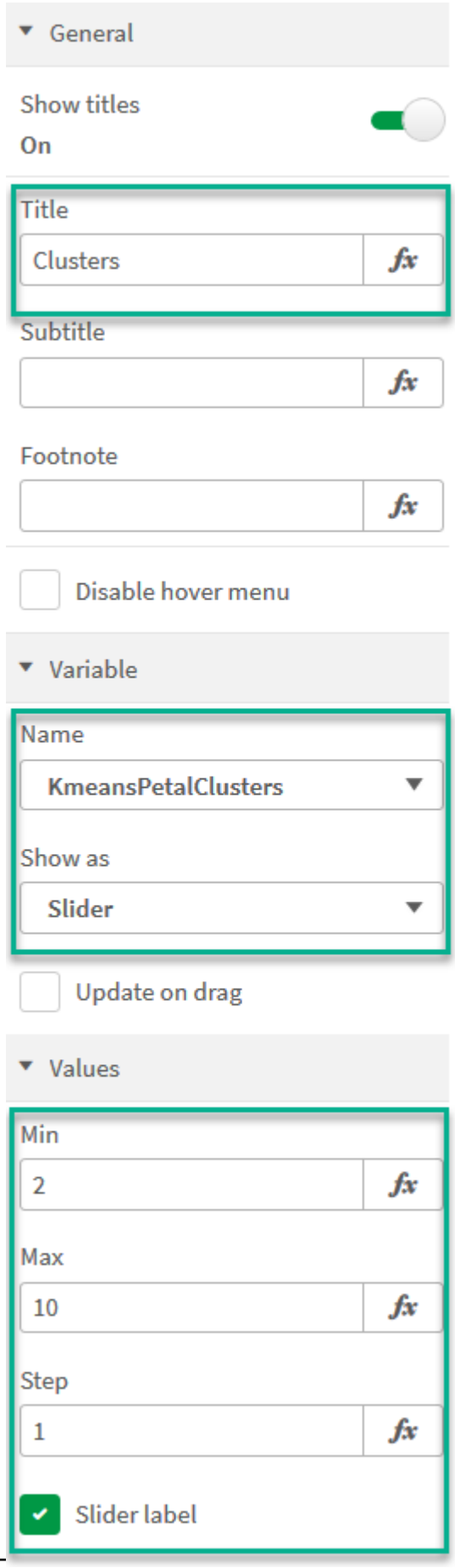
차트에서 두 개의 클러스터가 KMeans 표현식을 기준으로 색이 지정됩니다.
꽃잎(표현식 기준 색 지정) 차트에서 표현식을 기준으로 색이 지정된 클러스터



5. 클러스터 수에 대한 **변수 입력** 상자를 추가합니다.
    i. **자산** 패널의 **사용자 지정 개체**에서 **Qlik 대시보드 번들**을 선택합니다. 대시보드 번들에 액세스하지 않은 경우 생성한 변수를 사용하여 클러스터 수를 계속 변경하거나 표현식에서 직접 정수로 변경할 수 있습니다.
    ii. **변수 입력** 상자를 시트로 끕니다.
    iii. **모양**에서 **일반**을 클릭합니다.
    iv. **제목**에 대해 다음을 입력합니다. *클러스터*
    v. **변수**를 클릭합니다.
    vi. **이름**에 대해 다음 변수를 선택합니다. *KmeansPetalClusters*.
    vii. **표시**에 대해 **슬라이더**를 선택합니다.

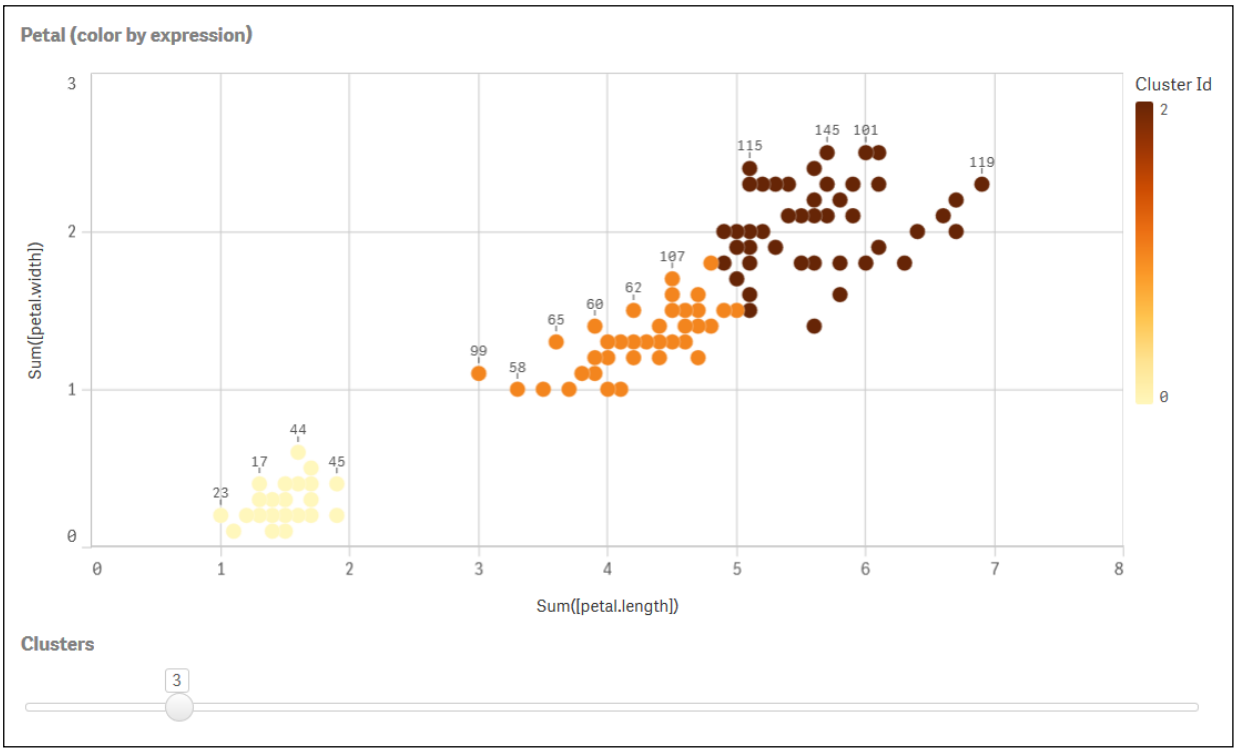
viii. **값**을 선택하고 필요에 따라 설정을 구성합니다.

클러스터 변수 입력 상자의 모양


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

편집이 완료되면 *Clusters* 변수 입력 상자에서 슬라이더를 사용하여 클러스터 수를 변경할 수 있습니다.

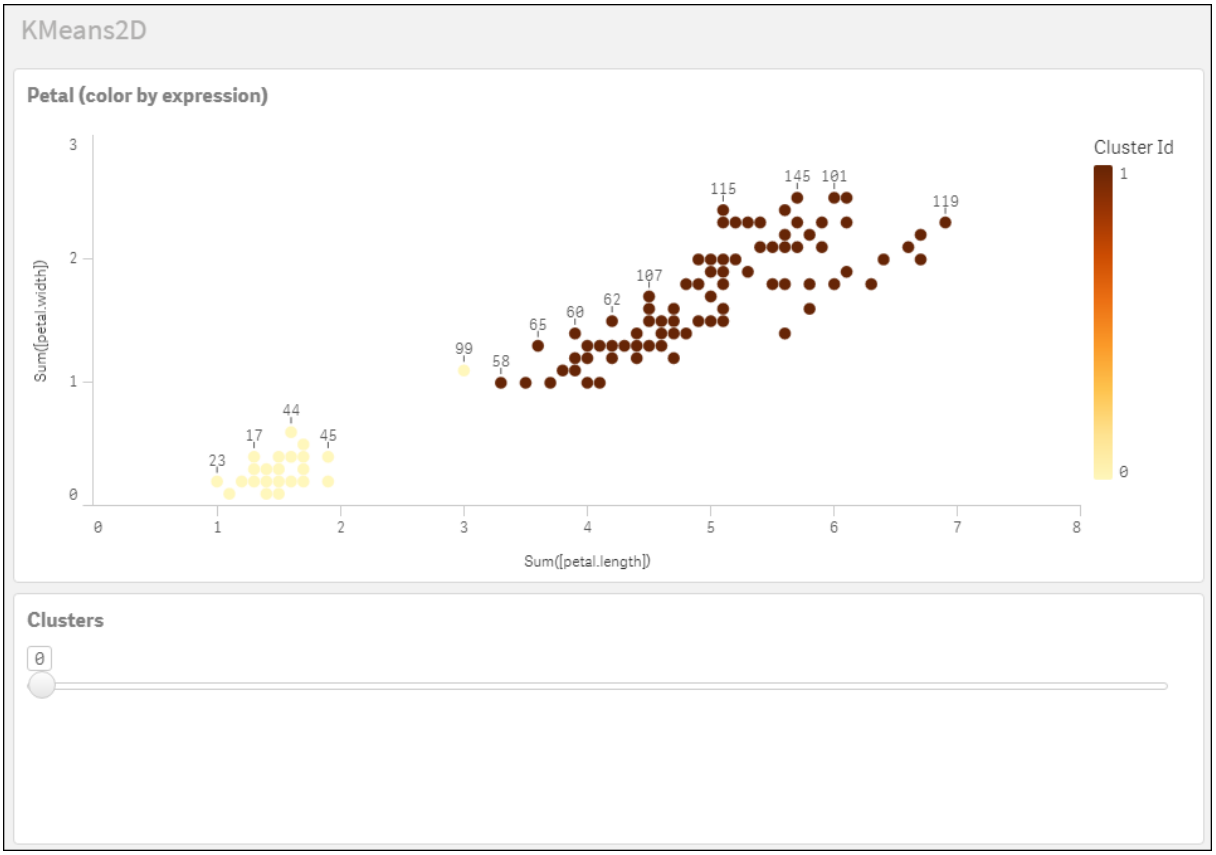
꽃잎(표현식 기준 색 지정) 차트에서 표현식을 기준으로 색이 지정된 클러스터



자동 클러스터링

KMeans 함수는 깊이 차(DeD)라는 방법을 사용하여 자동 클러스터링을 지원합니다. 사용자가 클러스터 수를 0으로 설정한 경우 해당 데이터 집합에 대한 최적의 클러스터 수가 결정됩니다. 클러스터 수의 정수(*k*)는 명시적으로 반환되지 않으며 KMeans 알고리즘 내에서 계산됩니다. 예를 들어 함수에서 0이 *KmeansPetalClusters* 값으로 지정되거나 변수 입력 상자를 통해 설정된 경우 최적의 클러스터 수를 기반으로 데이터 집합의 클러스터 할당이 자동으로 계산됩니다.

KMeans 깊이 차 메서드에 따라 (k)가 0으로 설정된 경우 최적의 클러스터 수가 결정됩니다.



Iris 데이터 집합: Qlik Sense에서 데이터 로드 편집기에 대한 인라인 로드

```
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 - 차트 함수

KMeansND()는 k-평균클러스터링을 적용하여 차트의 행을 평가하고 각 차트 행에 이 데이터 포인트가 할당된 클러스터의 클러스터 ID를 표시합니다. 클러스터링 알고리즘에서 사용되는 열은 매개 변수 coordinate_1, coordinate_2 등(최대 n열)에 의해 결정됩니다. 모두 집계 열입니다. 생성된 클러스터 수는 num_clusters 매개 변수에 의해 결정됩니다.

KMeansND는 데이터 포인트당 하나의 값을 반환합니다. 반환된 값은 이중 값이며 각 데이터 포인트가 할당된 클러스터에 해당하는 정수 값입니다.

구문:

```
KMeansND(num_clusters, num_iter, coordinate_1, coordinate_2 [,coordinate_3 [,...]])
```

반환 데이터 유형: dual

인수:

인수

인수	설명
num_clusters	클러스터 수를 지정하는 정수입니다.
num_iter	다시 초기화된 클러스터 중심을 사용한 클러스터링 반복 횟수입니다.
coordinate_1	첫 번째 좌표를 계산하는 집계는 일반적으로 차트로부터 만들 수 있는 스캐터 차트의 x 축입니다. 추가 매개 변수는 두 번째, 세 번째 및 네 번째 좌표 등을 계산합니다.

예: 차트 표현식
이 예제에서는 *Iris* 데이터 집합을 사용하여 스캐터 차트를 만든 다음 KMeans를 사용하여 표현식으로 데이터에 색을 지정합니다.

또한 *num_clusters* 인수에 대한 변수를 만든 다음 변수 입력 상자를 사용하여 클러스터 수를 변경합니다.

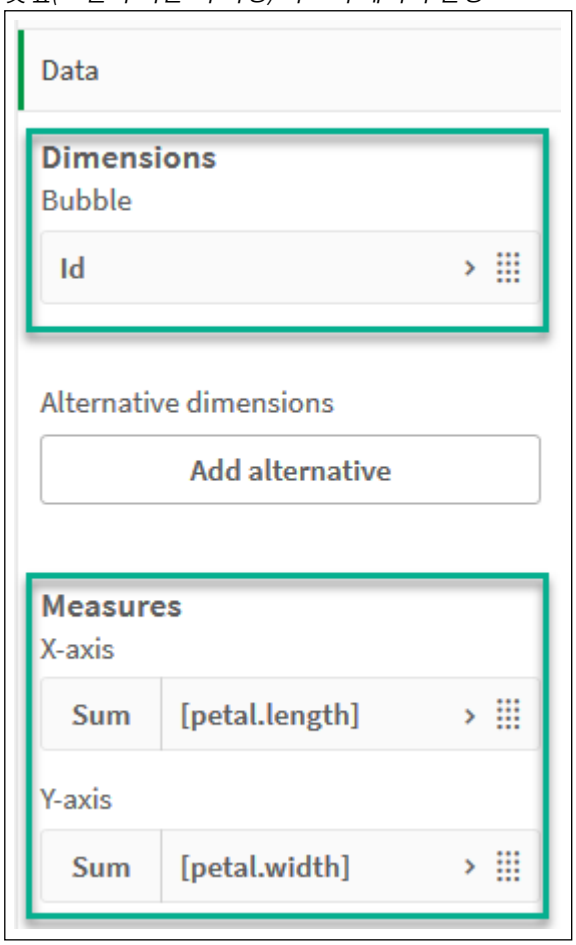
또한 *num_iter* 인수에 대한 변수를 만든 다음 두 번째 변수 입력 상자를 사용하여 반복 횟수를 변경합니다.

Iris 데이터 집합은 다양한 형식으로 공개적으로 사용할 수 있습니다. Qlik Sense의 데이터 로드 편집기를 사용하여 로드할 데이터를 인라인 테이블로 제공했습니다. 이 예에서는 데이터 테이블에 *Id* 열을 추가했습니다.

Qlik Sense에서 데이터 로드 후 다음을 수행합니다.

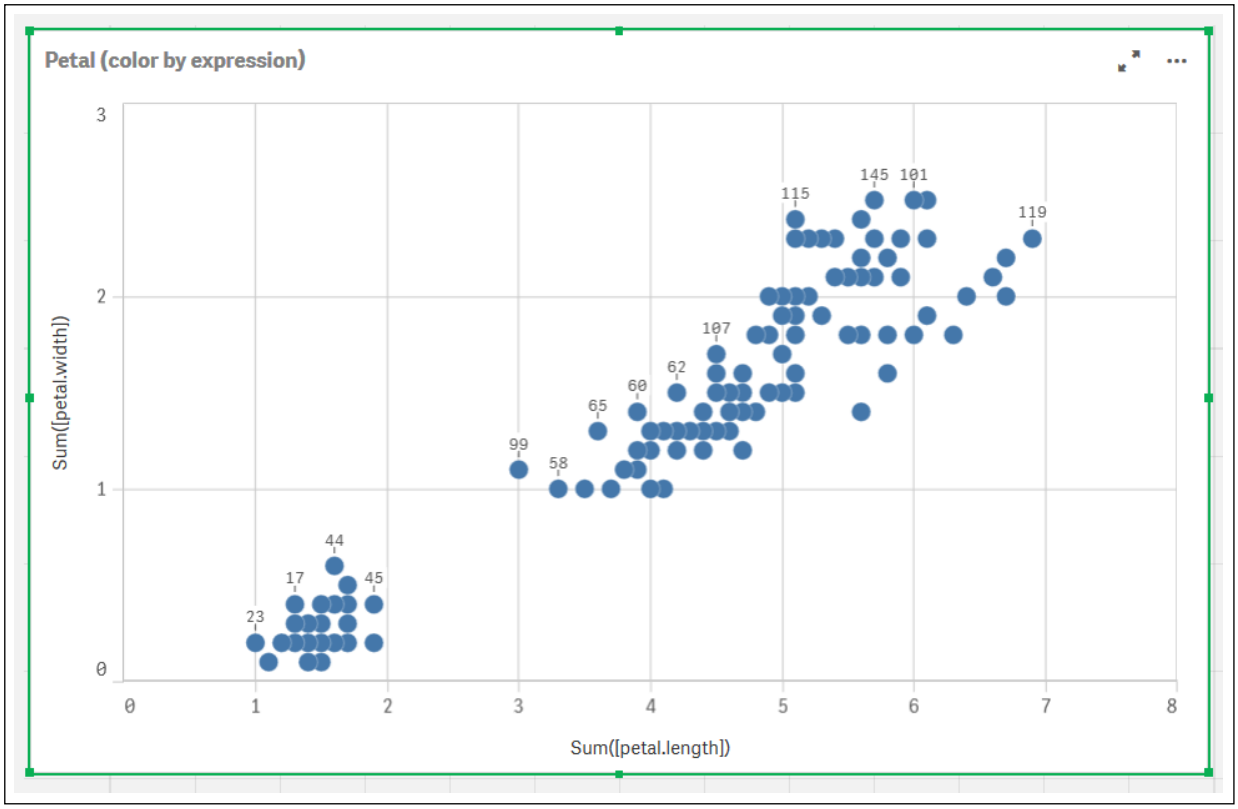
1. **스캐터 차트**를 새 시트로 끕니다. 차트의 이름을 *꽃잎(표현식으로 색 지정)*으로 지정합니다.
2. 변수를 만들어 클러스터 수를 지정합니다. 변수 **Name**에 *KmeansPetalClusters*를 입력합니다. 변수 **Definition**에 *=2*를 입력합니다.
3. 변수를 만들어 반복 횟수를 지정합니다. 변수 **Name**에 *KmeansNumberIterations*를 입력합니다. 변수 **Definition**에 *=1*을 입력합니다.
4. 차트에 대한 **데이터**를 구성합니다.
    i. **차원**에서 **거품** 필드에 대해 *id*를 선택합니다. 레이블에 대해 클러스터 Id를 입력합니다.
    ii. **측정값**에서 **X 축** 표현식에 대해 *Sum([petal.length])*를 선택합니다.
    iii. **측정값**에서 **Y 축** 표현식에 대해 *Sum([petal.width])*를 선택합니다.

꽃잎(표현식 기준 색 지정) 차트의 데이터 설정



데이터 포인트가 차트에 그려집니다.

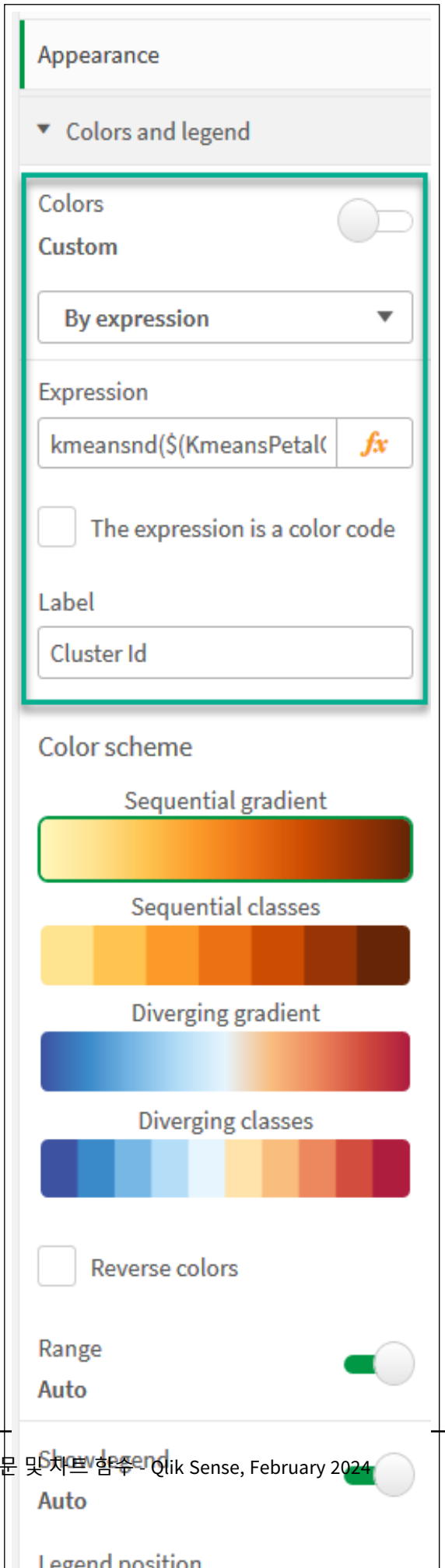
꽃잎(표현식 기준 색 지정) 차트의 데이터 포인트



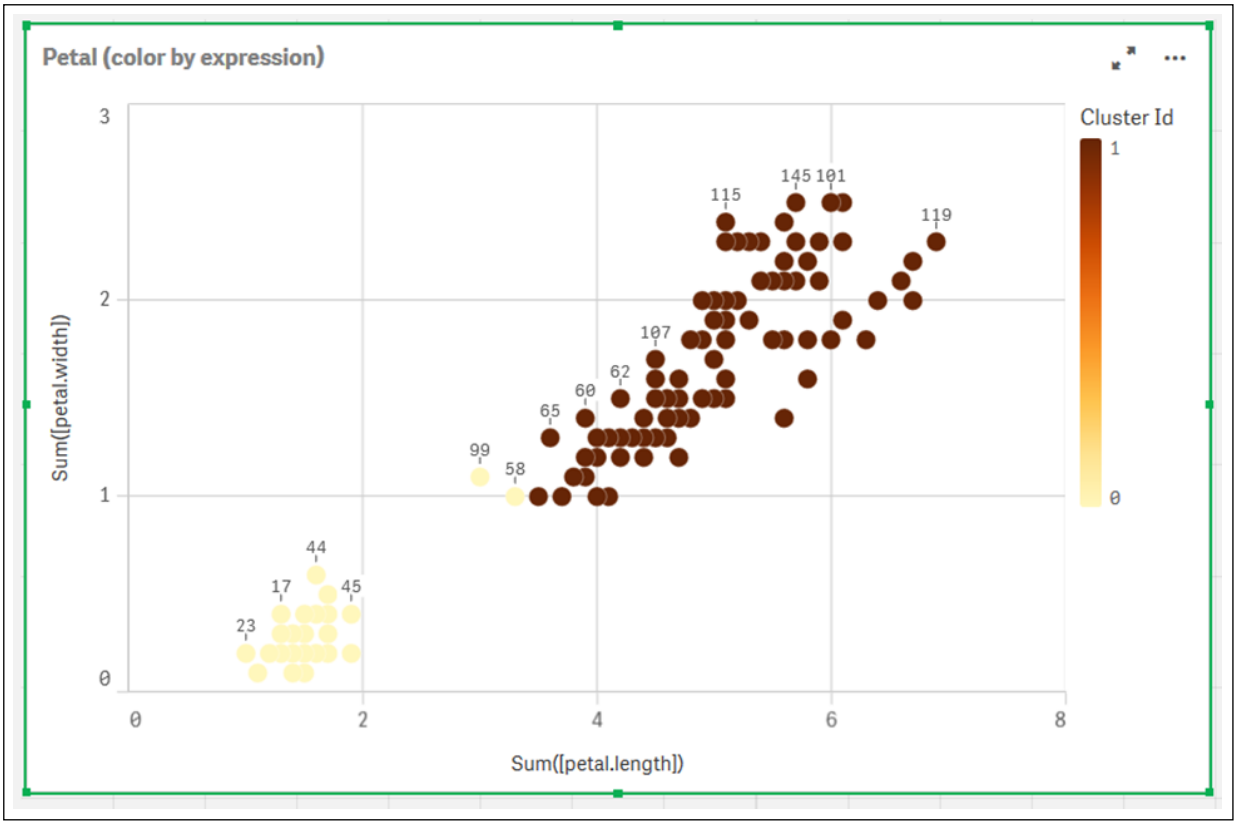
5. 차트의 **모양**을 구성합니다.
   i. **색 및 범례**에서 **색**에 대해 **사용자 지정**을 선택합니다.
   ii. **표현식 기준**으로 차트 색을 지정하도록 선택합니다.
   iii. **표현식**에 대해 다음을 입력합니다. *kmeansnd ($(KmeansPetalClusters),$(KmeansNumberIterations), Sum([petal.length]), Sum ([petal.width]),Sum([sepal.length]), Sum([sepal.width]))*
   *KmeansPetalClusters*는 *2*로 설정한 변수입니다. *KmeansNumberIterations*는 *1*로 설정한 변수입니다.
   또는 다음을 입력합니다. *kmeansnd(2, 2, Sum([petal.length]), Sum([petal.width]),Sum ([sepal.length]), Sum([sepal.width]))*
   iv. **표현식은 색상 코드입니다** 확인란을 선택 취소합니다.

v. **레이블**에 대해 다음을 입력합니다. *클러스터 Id*

꽃잎(표현식 기준 색 지정) 차트의 모양 설정


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

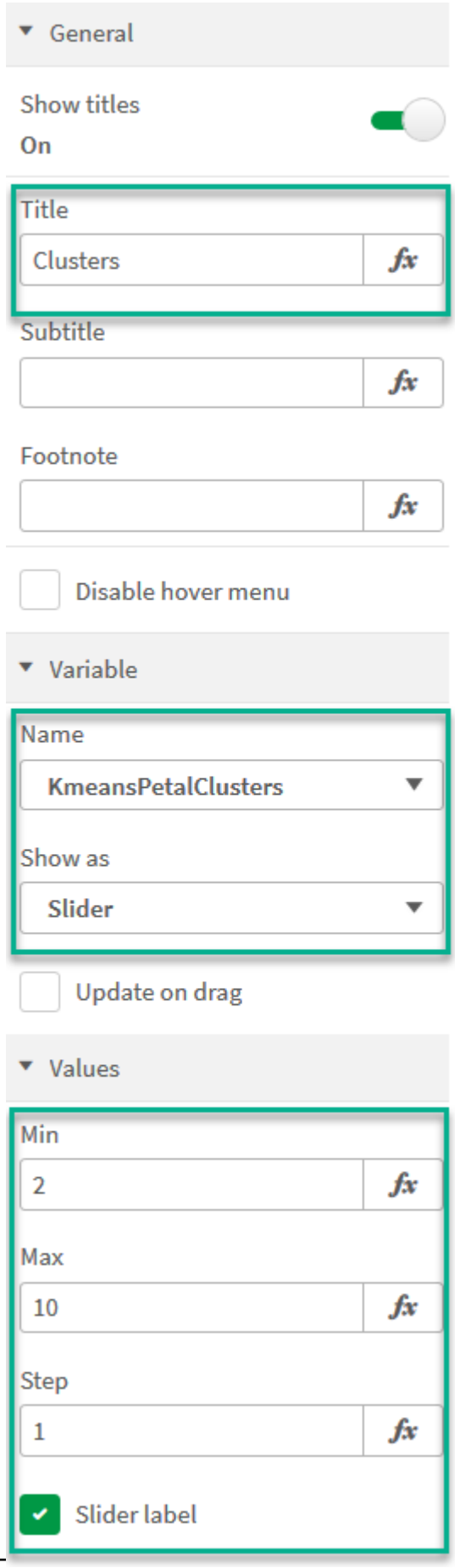
차트에서 두 개의 클러스터가 KMeans 표현식을 기준으로 색이 지정됩니다.
꽃잎(표현식 기준 색 지정) 차트에서 표현식을 기준으로 색이 지정된 클러스터



6. 클러스터 수에 대한 **변수 입력** 상자를 추가합니다.
    i. **자산** 패널의 **사용자 지정 개체**에서 **Qlik 대시보드 번들**을 선택합니다. 대시보드 번들에 액세스하지 않은 경우 생성한 변수를 사용하여 클러스터 수를 계속 변경하거나 표현식에서 직접 정수로 변경할 수 있습니다.
    ii. **변수 입력** 상자를 시트로 끕니다.
    iii. **모양**에서 **일반**을 클릭합니다.
    iv. **제목**에 대해 다음을 입력합니다. *클러스터*
    v. **변수**를 클릭합니다.
    vi. **이름**에 대해 다음 변수를 선택합니다. *KmeansPetalClusters*.
    vii. **표시**에 대해 **슬라이더**를 선택합니다.

viii. **값**을 선택하고 필요에 따라 설정을 구성합니다.

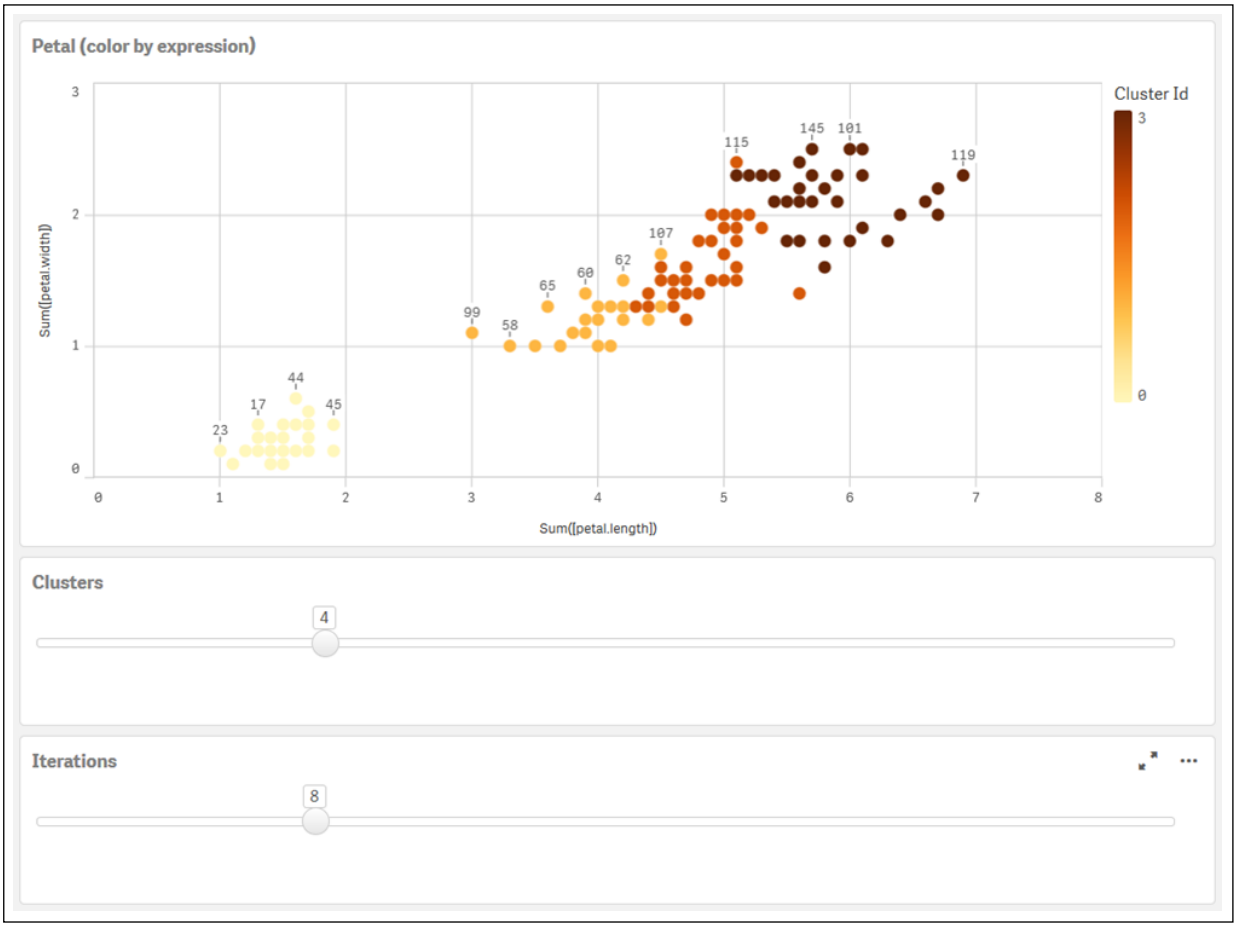
클러스터 변수 입력 상자의 모양


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. 반복 횟수에 대한 **변수 입력** 상자를 추가합니다.
    i. **변수 입력** 상자를 시트로 끕니다.
    ii. **모양**에서 **일반**을 선택합니다.
    iii. **제목**에 대해 다음을 입력합니다. *반복*
    iv. **모양**에서 **변수**를 선택합니다.
    v. **이름**에서 다음 변수를 선택합니다. *KmeansNumberIterations*.
    vi. 필요에 따라 추가 설정을 구성합니다.

이제 변수 입력 상자에서 슬라이더를 사용하여 클러스터 수 및 반복 횟수를 변경할 수 있습니다.

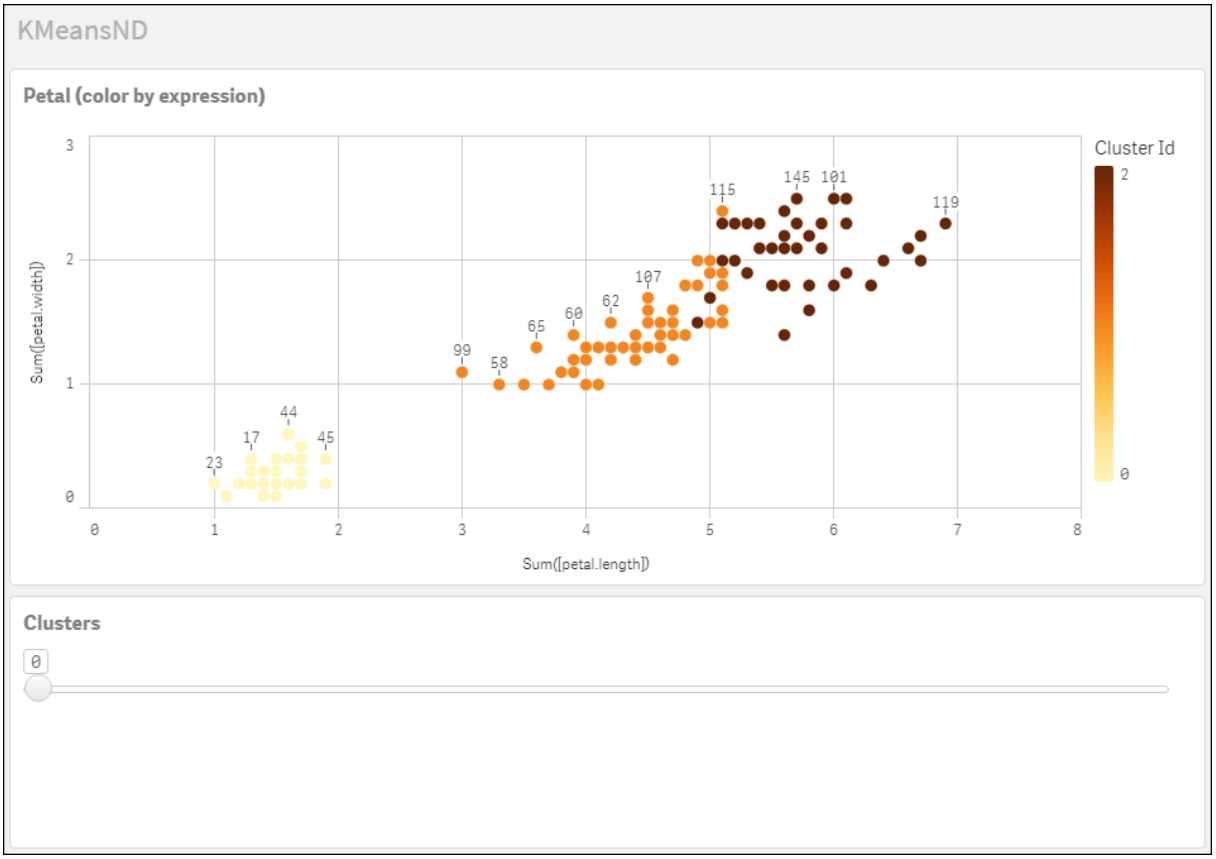
꽃잎(표현식 기준 색 지정) 차트에서 표현식을 기준으로 색이 지정된 클러스터



자동 클러스터링

KMeans 함수는 깊이 차(DeD)라는 방법을 사용하여 자동 클러스터링을 지원합니다. 사용자가 클러스터 수를 0으로 설정한 경우 해당 데이터 집합에 대한 최적의 클러스터 수가 결정됩니다. 클러스터 수의 정수(*k*)는 명시적으로 반환되지 않으며 KMeans 알고리즘 내에서 계산됩니다. 예를 들어 함수에서 0이 *KmeansPetalClusters* 값으로 지정되거나 변수 입력 상자를 통해 설정된 경우 최적의 클러스터 수를 기반으로 데이터 집합의 클러스터 할당이 자동으로 계산됩니다. Iris 데이터 집합에서 클러스터 수로 0이 선택된 경우 알고리즘에 따라 이 데이터 집합의 최적의 클러스터 수(3)가 결정됩니다(자동 클러스트링).

KMeans 깊이 차 메서드에 따라 (k)가 0으로 설정된 경우 최적의 클러스터 수가 결정됩니다.



Iris 데이터 집합: Qlik Sense에서 데이터 로드 편집기에 대한 인라인 로드

```
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 - 차트 함수

KMeansCentroid2D()는 k-평균클러스터링을 적용하여 차트의 행을 평가하고 각 차트 행에 이 데이터 포인트가 할당된 클러스터의 원하는 좌표를 표시합니다. 클러스터링 알고리즘에서 사용되는 열은 매개 변수 coordinate_1 및 coordinate_2에 의해 각각 결정됩니다. 둘 다 집계 열입니다. 생성된 클러스터 수는 num_clusters 매개 변수에 의해 결정됩니다. 데이터는 선택적으로 표준 매개 변수로 정규화할 수 있습니다.

KMeansCentroid2D는 데이터 포인트당 하나의 값을 반환합니다. 반환된 값은 이중 값이며 데이터 포인트가 할당된 클러스터 중심에 해당하는 위치의 좌표 중 하나입니다.

구문:

```
KMeansCentroid2D(num_clusters, coordinate_no, coordinate_1, coordinate_2 [, norm])
```

반환 데이터 유형: dual

인수:

인수

인수	설명
num_clusters	클러스터 수를 지정하는 정수입니다.
coordinate_no	원하는 중심의 좌표 수(예: x 축, y 축 또는 z 축에 해당)입니다.
coordinate_1	첫 번째 좌표를 계산하는 집계는 일반적으로 차트로부터 만들 수 있는 스캐터 차트의 x 축입니다. 추가 매개 변수 coordinate_2는 두 번째 좌표를 계산합니다.

인수	설명
norm	선택적 정규화 방법이 KMeans 클러스터링 전에 데이터 집합에 적용됩니다. 가능한 값 : 정규화가 없는 경우 0 또는 '없음' z 점수 정규화의 경우 1 또는 'zscore' 최소-최대 정규화의 경우 2 또는 'minmax' 매개 변수가 제공되지 않거나 제공된 매개 변수가 잘못된 경우 정규화가 적용되지 않습니다. Z 점수는 기능 평균과 표준 편차를 기준으로 데이터를 정규화합니다. Z 점수는 각 기능이 동일한 척도를 갖도록 하지 않지만 이상값을 처리할 때 최소-최대보다 더 나은 접근 방식입니다. 최소-최대 정규화는 각각의 최솟값과 최댓값을 가져오고 각 데이터 포인트를 다시 계산하여 특성이 동일한 척도를 갖도록 합니다.

자동 클러스터링

KMeans 함수는 깊이 차(DeD)라는 방법을 사용하여 자동 클러스터링을 지원합니다. 사용자가 클러스터 수를 0으로 설정한 경우 해당 데이터 집합에 대한 최적의 클러스터 수가 결정됩니다. 클러스터 수의 정수(*k*)는 명시적으로 반환되지 않으며 KMeans 알고리즘 내에서 계산됩니다. 예를 들어 함수에서 0이 *KmeansPetalClusters* 값으로 지정되거나 변수 입력 상자를 통해 설정된 경우 최적의 클러스터 수를 기반으로 데이터 집합의 클러스터 할당이 자동으로 계산됩니다.

KMeansCentroidND - 차트 함수

KMeansCentroidND()는 k-평균클러스터링을 적용하여 차트의 행을 평가하고 각 차트 행에 이 데이터 포인트가 할당된 클러스터의 원하는 좌표를 표시합니다. 클러스터링 알고리즘에서 사용되는 열은 매개 변수 coordinate_1, coordinate_2 등(최대 n열)에 의해 결정됩니다. 모두 집계 열입니다. 생성된 클러스터 수는 num_clusters 매개 변수에 의해 결정됩니다.

KMeansCentroidND는 행당 하나의 값을 반환합니다. 반환된 값은 이중 값이며 데이터 포인트가 할당된 클러스터 중심에 해당하는 위치의 좌표 중 하나입니다.

구문:

```
KMeansCentroidND((num_clusters, num_iter, coordinate_no, coordinate_1,
coordinate_2 [,coordinate_3 [, ...]])
```

반환 데이터 유형: dual

인수:

인수

인수	설명
num_clusters	클러스터 수를 지정하는 정수입니다.
num_iter	다시 초기화된 클러스터 중심을 사용한 클러스터링 반복 횟수입니다.
coordinate_no	원하는 중심의 좌표 수(예: x 축, y 축 또는 z 축에 해당)입니다.
coordinate_1	첫 번째 좌표를 계산하는 집계는 일반적으로 차트로부터 만들 수 있는 스캐터 차트의 x 축입니다. 추가 매개 변수는 두 번째, 세 번째 및 네 번째 좌표 등을 계산합니다.

자동 클러스터링

KMeans 함수는 깊이 차(DeD)라는 방법을 사용하여 자동 클러스터링을 지원합니다. 사용자가 클러스터 수를 0으로 설정한 경우 해당 데이터 집합에 대한 최적의 클러스터 수가 결정됩니다. 클러스터 수의 정수(*k*)는 명시적으로 반환되지 않으며 KMeans 알고리즘 내에서 계산됩니다. 예를 들어 함수에서 0이 *KmeansPetalClusters* 값으로 지정되거나 변수 입력 상자를 통해 설정된 경우 최적의 클러스터 수를 기반으로 데이터 집합의 클러스터 할당이 자동으로 계산됩니다.

STL_Trend - 차트 함수

STL_Trend는 시계열 분해 함수입니다. **STL_Seasonal** 및 **STL_Residual**과 함께 이 함수는 시계열을 계절성, 추세 및 잔차 구성 요소로 분해하는 데 사용됩니다. STL 알고리즘 컨텍스트에서 시계열 분해는 입력 메트릭 및 기타 매개 변수가 주어지면 반복되는 계절성 패턴과 일반적인 추세를 식별하는 데 사용됩니다. **STL_Trend** 함수는 시계열 데이터에서 계절성 패턴이나 주기와 관계없이 일반적인 추세를 식별합니다.

세 가지 STL 함수는 간단한 합계를 통해 입력 메트릭과 관련되어 있습니다.

STL_Trend + **STL_Seasonal** + **STL_Residual** = 입력 메트릭

STL(Loess를 사용한 계절성 및 추세 분해)은 데이터 스무딩 기술을 사용하며 입력 매개 변수를 통해 사용자가 수행하는 계산의 주기성을 조정할 수 있습니다. 이 주기성은 분석에서 입력 메트릭(측정값)의 시간 차원이 분할되는 방식을 결정합니다.

최소한 **STL_Trend**는 입력 메트릭(`target_measure`)과 `period_int`에 대한 정수 값을 가져와 부동 소수점 값을 반환합니다. 입력 메트릭은 시간 차원에 따라 달라지는 집계 형식입니다. 선택적으로 매끄러운 알고리즘을 조정하기 위해 `seasonal_smoother` 및 `trend_smoother`에 대한 값을 포함할 수 있습니다.

차트의 식 편집기에 직접 입력하거나 측정값에 **시계열 분해** 수정자를 추가하여 이 함수를 사용할 수 있습니다. 통찰력에서 분석 유형으로 사용할 수도 있습니다.

구문:

```
STL_Trend(target_measure, period_int [,seasonal_smoother [,trend_smoother]])
```

반환 데이터 유형: dual

인수

인수	설명
target_measure	계절성 및 추세 구성 요소로 분해할 측정값입니다. 이는 시간 차원에 따라 변하는 Sum(Sales) 또는 Sum(Passengers)와 같은 측정값이어야 합니다. 이는 상수 값이 아니어야 합니다.
period_int	데이터 집합의 주기성입니다. 이 매개 변수는 신호의 한 주기 또는 계절성 주기를 구성하는 불연속 단계의 수를 나타내는 정수 값입니다. 예를 들어 시계열이 연도의 각 분기에 대해 하나의 섹션으로 분할되는 경우 주기성을 연도로 정의하려면 **period_int** 값을 4로 설정해야 합니다.
seasonal_smoother	계절성 스무더의 길이. 이 값은 홀수 정수여야 합니다. 계절성 스무더는 여러 기간에 걸쳐 계절성 변동의 특정 단계에 대한 데이터를 사용합니다. 각 기간에서 시간 차원의 하나의 불연속 단계가 사용됩니다. 계절성 스무더는 스무딩에 사용된 기간 수를 나타냅니다. 예를 들어 시간 차원이 월별로 분할되고 기간이 연도(12)인 경우 계절성 구성 요소가 계산되므로 각 연도의 특정 월이 해당 연도와 인접 연도 모두에서 같은 달의 데이터에서 계산됩니다. **seasonal_smoother** 값은 스무딩에 사용된 년 수입니다.
trend_smoother	추세 스무더의 길이. 이 값은 홀수 정수여야 합니다. 추세 스무더는 **period_int** 매개 변수와 동일한 시간 배율을 사용하며 해당 값은 스무딩에 사용되는 과립의 수입니다. 예를 들어 시계열이 월별로 분할된 경우 추세 스무더는 스무딩에 사용된 월 수가 됩니다.

STL_Trend 차트 함수는 다음 함수와 함께 사용되는 경우가 많습니다.

관련 함수

함수	상호 작용
STL_Seasonal - 차트 함수 (page 1371)	시계열의 계절성 구성 요소를 계산하는 데 사용되는 함수입니다.

함수	상호 작용
STL_Residual - 차트 함수 (page 1373)	입력 메트릭을 계절성 및 추세 구성 요소로 나눌 때 측정값 변동의 일부가 두 가지 주요 구성 요소에 맞지 않습니다. **STL_Residual** 함수는 분해의 이 부분을 계산합니다.

이 함수를 사용하는 방법을 보여 주는 전체 예가 포함된 자습서는 *자습서 - Qlik Sense의 시계열 분해 (page 1375)*을 참조하십시오.

STL_Seasonal - 차트 함수

STL_Seasonal은 시계열 분해 함수입니다. **STL_Trend** 및 **STL_Residual**과 함께 이 함수는 시계열을 계절성, 추세 및 잔차 구성 요소로 분해하는 데 사용됩니다. STL 알고리즘 컨텍스트에서 시계열 분해는 입력 메트릭 및 기타 매개 변수가 주어지면 반복되는 계절성 패턴과 일반적인 추세를 식별하는 데 사용됩니다. **STL_Seasonal** 함수는 시계열 내에서 계절성 패턴을 식별하여 데이터에 표시되는 일반적인 추세와 구분할 수 있습니다.

세 가지 STL 함수는 간단한 합계를 통해 입력 메트릭과 관련되어 있습니다.

STL_Trend + **STL_Seasonal** + **STL_Residual** = 입력 메트릭

STL(Loess를 사용한 계절성 및 추세 분해)은 데이터 스무딩 기술을 사용하며 입력 매개 변수를 통해 사용자가 수행하는 계산의 주기성을 조정할 수 있습니다. 이 주기성은 분석에서 입력 메트릭(측정값)의 시간 차원이 분할되는 방식을 결정합니다.

최소한 **STL_Seasonal**은 입력 메트릭(`target_measure`)과 `period_int`에 대한 정수 값을 가져와 부동 소수점 값을 반환합니다. 입력 메트릭은 시간 차원에 따라 달라지는 집계 형식입니다. 선택적으로 매끄러운 알고리즘을 조정하기 위해 `seasonal_smoother` 및 `trend_smoother`에 대한 값을 포함할 수 있습니다.

차트의 식 편집기에 직접 입력하거나 측정값에 **시계열 분해** 수정자를 추가하여 이 함수를 사용할 수 있습니다. 통찰력에서 분석 유형으로 사용할 수도 있습니다.

구문:

```
STL_Seasonal(target_measure, period_int [,seasonal_smoother [,trend_
smoother]])
```

반환 데이터 유형: dual

인수

인수	설명
target_measure	계절성 및 추세 구성 요소로 분해할 측정값입니다. 이는 시간 차원에 따라 변하는 Sum(Sales) 또는 Sum(Passengers)와 같은 측정값이어야 합니다. 이는 상수 값이 아니어야 합니다.
period_int	데이터 집합의 주기성입니다. 이 매개 변수는 신호의 한 주기 또는 계절성 주기를 구성하는 불연속 단계의 수를 나타내는 정수 값입니다. 예를 들어 시계열이 연도의 각 분기에 대해 하나의 섹션으로 분할되는 경우 주기성을 연도로 정의하려면 **period_int** 값을 4로 설정해야 합니다.
seasonal_smoother	계절성 스무더의 길이. 이 값은 홀수 정수여야 합니다. 계절성 스무더는 여러 기간에 걸쳐 계절성 변동의 특정 단계에 대한 데이터를 사용합니다. 각 기간에서 시간 차원의 하나의 불연속 단계가 사용됩니다. 계절성 스무더는 스무딩에 사용된 기간 수를 나타냅니다. 예를 들어 시간 차원이 월별로 분할되고 기간이 연도(12)인 경우 계절성 구성 요소가 계산되므로 각 연도의 특정 월이 해당 연도와 인접 연도 모두에서 같은 달의 데이터에서 계산됩니다. **seasonal_smoother** 값은 스무딩에 사용된 년 수입니다.
trend_smoother	추세 스무더의 길이. 이 값은 홀수 정수여야 합니다. 추세 스무더는 **period_int** 매개 변수와 동일한 시간 배율을 사용하며 해당 값은 스무딩에 사용되는 과립의 수입니다. 예를 들어 시계열이 월별로 분할된 경우 추세 스무더는 스무딩에 사용된 월 수가 됩니다.

STL_Seasonal 차트 함수는 다음 함수와 함께 사용되는 경우가 많습니다.

관련 함수

함수	상호 작용
STL_Trend - 차트 함수 (page 1369)	시계열의 추세 구성 요소를 계산하는 데 사용되는 함수입니다.

함수	상호 작용
STL_Residual - 차트 함수 (page 1373)	입력 메트릭을 계절성 및 추세 구성 요소로 나눌 때 측정값 변동의 일부가 두 가지 주요 구성 요소에 맞지 않습니다. **STL_Residual** 함수는 분해의 이 부분을 계산합니다.

이 함수를 사용하는 방법을 보여 주는 전체 예가 포함된 자습서는 *자습서 - Qlik Sense의 시계열 분해 (page 1375)*을 참조하십시오.

STL_Residual - 차트 함수

STL_Residual은 시계열 분해 함수입니다. **STL_Seasonal** 및 **STL_Trend**와 함께 이 함수는 시계열을 계절성, 추세 및 잔차 구성 요소로 분해하는 데 사용됩니다. STL 알고리즘 컨텍스트에서 시계열 분해는 입력 메트릭 및 기타 매개 변수가 주어지면 반복되는 계절성 패턴과 일반적인 추세를 식별하는 데 사용됩니다. 이 작업을 수행할 때 입력 메트릭의 변동 중 일부는 계절성 구성 요소나 추세 구성 요소에 맞지 않으며 잔여 구성 요소로 정의됩니다. **STL_Residual** 차트 함수는 계산의 이 부분을 캡처합니다.

세 가지 STL 함수는 간단한 합계를 통해 입력 메트릭과 관련되어 있습니다.

STL_Trend + **STL_Seasonal** + **STL_Residual** = 입력 메트릭

STL(Loess를 사용한 계절성 및 추세 분해)은 데이터 스무딩 기술을 사용하며 입력 매개 변수를 통해 사용자가 수행하는 계산의 주기성을 조정할 수 있습니다. 이 주기성은 분석에서 입력 메트릭(측정값)의 시간 차원이 분할되는 방식을 결정합니다.

시계열 분해는 주로 데이터의 계절성과 일반적인 변동을 찾기 때문에 잔여의 정보는 세 가지 구성 요소 중 가장 덜 중요한 것으로 간주됩니다. 그러나 왜곡되거나 주기적인 잔여 구성 요소는 잘못된 주기 설정과 같은 계산 문제를 식별하는 데 도움이 될 수 있습니다.

최소한 **STL_Residual**은 입력 메트릭(`target_measure`)과 해당 `period_int`에 대한 정수 값을 사용하여 부동 소수점 값을 반환합니다. 입력 메트릭은 시간 차원에 따라 달라지는 집계 형식입니다. 선택적으로 매끄러운 알고리즘을 조정하기 위해 `seasonal_smoother` 및 `trend_smoother`에 대한 값을 포함할 수 있습니다.

차트의 식 편집기에 직접 입력하거나 측정값에 **시계열 분해** 수정자를 추가하여 이 함수를 사용할 수 있습니다. 통찰력에서 분석 유형으로 사용할 수도 있습니다.

구문:

```
STL_Residual(target_measure, period_int [,seasonal_smoother [,trend_
smoother]])
```

반환 데이터 유형: dual

인수

인수	설명
target_measure	계절성 및 추세 구성 요소로 분해할 측정값입니다. 이는 시간 차원에 따라 변하는 Sum(Sales) 또는 Sum(Passengers)와 같은 측정값이어야 합니다. 이는 상수 값이 아니어야 합니다.
period_int	데이터 집합의 주기성입니다. 이 매개 변수는 신호의 한 주기 또는 계절성 주기를 구성하는 불연속 단계의 수를 나타내는 정수 값입니다. 예를 들어 시계열이 연도의 각 분기에 대해 하나의 섹션으로 분할되는 경우 주기성을 연도로 정의하려면 **period_int** 값을 4로 설정해야 합니다.
seasonal_smoother	계절성 스무더의 길이. 이 값은 홀수 정수여야 합니다. 계절성 스무더는 여러 기간에 걸쳐 계절성 변동의 특정 단계에 대한 데이터를 사용합니다. 각 기간에서 시간 차원의 하나의 불연속 단계가 사용됩니다. 계절성 스무더는 스무딩에 사용된 기간 수를 나타냅니다. 예를 들어 시간 차원이 월별로 분할되고 기간이 연도(12)인 경우 계절성 구성 요소가 계산되므로 각 연도의 특정 월이 해당 연도와 인접 연도 모두에서 같은 달의 데이터에서 계산됩니다. **seasonal_smoother** 값은 스무딩에 사용된 년 수입니다.
trend_smoother	추세 스무더의 길이. 이 값은 홀수 정수여야 합니다. 추세 스무더는 **period_int** 매개 변수와 동일한 시간 배율을 사용하며 해당 값은 스무딩에 사용되는 과립의 수입니다. 예를 들어 시계열이 월별로 분할된 경우 추세 스무더는 스무딩에 사용된 월 수가 됩니다.

STL_Residual 차트 함수는 다음 함수와 함께 사용되는 경우가 많습니다.

관련 함수

함수	상호 작용
STL_Seasonal - 차트 함수 (page 1371)	시계열의 계절성 구성 요소를 계산하는 데 사용되는 함수입니다.
STL_Trend - 차트 함수 (page 1369)	시계열의 추세 구성 요소를 계산하는 데 사용되는 함수입니다.

이 함수를 사용하는 방법을 보여 주는 전체 예가 포함된 자습서는 *자습서 - Qlik Sense의 시계열 분해 (page 1375)*을 참조하십시오.

자습서 - Qlik Sense의 시계열 분해

이 자습서에서는 세 가지 차트 함수를 사용하여 STL 알고리즘을 사용하여 시계열을 분해하는 방법을 보여 줍니다.

이 자습서는 STL 알고리즘의 기능을 보여 주기 위해 월간 항공사를 이용하는 승객 수에 대한 시계열 데이터를 사용합니다. **STL_Trend**, **STL_Seasonal** 및 **STL_Residual** 차트 함수는 시각화를 만드는 데 사용됩니다. Qlik Sense의 시계열 분해에 대한 자세한 내용은 *시계열 분해 함수 (page 1323)*를 참조하십시오.

앱 만들기

새 앱을 만들고 데이터 집합을 가져오는 것으로 시작합니다.

이 데이터 집합 다운로드:

자습서 - 시계열 분해

이 파일에는 항공사의 월간 승객 수에 관한 데이터가 포함되어 있습니다.

다음과 같이 하십시오.

1. 허브에서 **새 앱 만들기**를 클릭합니다.
2. 앱을 열고 *Tutorial - Time series decomposition.csv* 파일을 여기에 드롭합니다.

데이터 준비 및 로드

Qlik Sense가 YearMonth 필드를 올바르게 해석하려면 데이터 관리자를 사용하여 필드를 문자열 값이 있는 필드가 아닌 날짜 필드로 인식해야 할 수 있습니다. 일반적으로 이 단계는 자동으로 처리되지만 이 경우 날짜는 약간 특이한 *YYYY-MM* 서식으로 표시됩니다.

1. 데이터 관리자에서 테이블을 선택하고을 클릭합니다.
2. *YearMonth* 필드를 선택한 상태에서을 클릭하고 **필드 유형**을 **날짜**로 설정합니다.
3. **입력 형식**에서 *YYYY-MM*을 입력합니다.
4. **표시 형식**에서 *YYYY-MM*을 입력하고 **확인**을 클릭합니다.
   이제 필드에 캘린더 아이콘이 표시됩니다.
5. **데이터 로드**를 클릭합니다.

이제 STL 함수를 사용하여 데이터를 시각적으로 표시할 준비가 되었습니다.

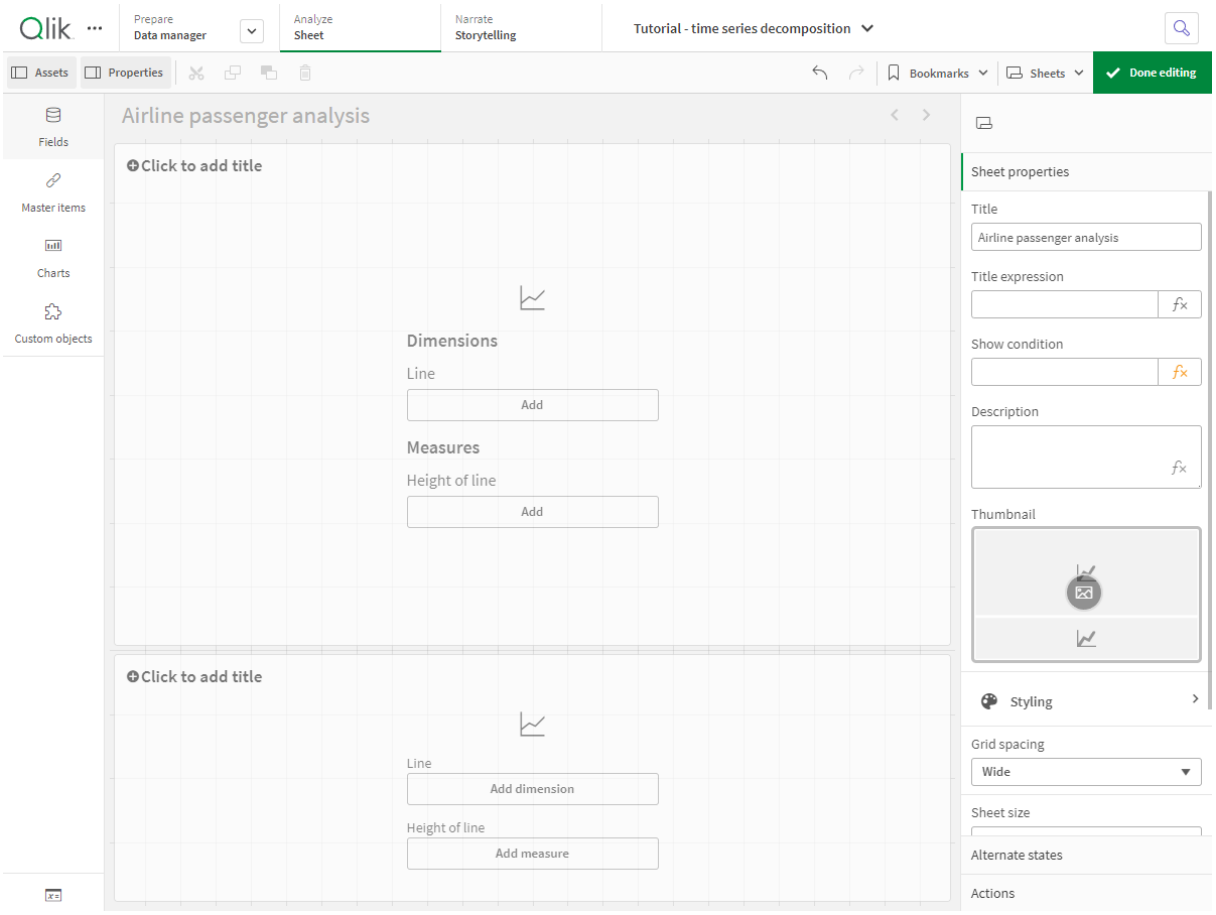
시각화 만들기

다음으로 **STL_Trend**, **STL_Seasonal** 및 **STL_Residual** 차트 기능의 기능을 보여 주는 두 개의 꺾은선형 차트를 만듭니다.

새 시트를 열고 제목을 지정합니다.

시트에 두 개의 꺾은선형 차트를 추가합니다. 다음 이미지와 일치하도록 차트의 크기를 조정하고 위치를 변경합니다.

빈 앱 시트의 Qlik Sense 그리드 윤곽선



첫 번째 꺾은선형 차트: 추세 및 계절성 구성 요소

다음과 같이 하십시오.

1. 첫 번째 꺾은선형 차트에 제목 *계절성 및 추세*를 추가합니다.
2. *YearMonth*를 차원으로 추가하고 *날짜*라는 레이블을 지정합니다.
3. 다음 측정값을 추가하고 *월 승객 수*라는 레이블을 지정합니다.
   =Sum(Passengers)
4. **데이터**에서 *월 승객 수* 측정값을 펼치고 **추세선 추가**를 클릭합니다.
5. **유형**을 **선형**으로 설정합니다.
   이 추세선을 추세 구성 요소의 매끄럽게 된 출력과 비교합니다.
6. 다음 측정값을 추가하여 추세 구성 요소를 표시하고 *추세* 레이블을 지정합니다.
   =STL_Trend(SUM(Passengers), 12)
7. 다음으로 다음 측정값을 추가하여 계절성 구성 요소를 표시하고 *계절성*이라는 레이블을 지정합니다.
   =STL_Trend(SUM(Passengers), 12)
8. **모양** > **프레젠테이션**, **스크롤 막대**를 **없음**으로 설정합니다.
9. 기본 색을 유지하거나 기본 설정에 맞게 변경합니다.

두 번째 꺾은선형 차트: 잔차 구성 요소

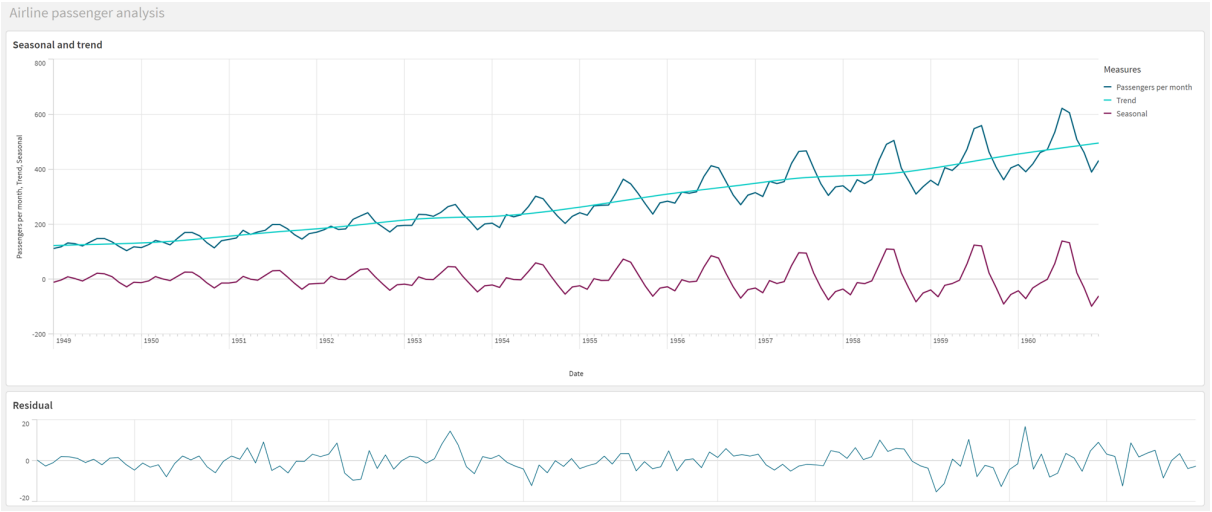
다음으로 두 번째 꺾은선형 차트를 구성합니다. 이 시각화는 시계열의 잔차 구성 요소를 표시합니다.

다음과 같이 하십시오.

1. 꺾은선형 차트를 시트로 끕니다. 제목 *잔차*을 추가합니다.
2. 차원으로 *날짜*를 추가합니다.
3. 다음 측정값을 추가하고 *잔차*라는 레이블을 지정합니다.
   =STL_Residual(SUM(Passengers), 12)
4. **모양** > **프레젠테이션**, **스크롤 막대**를 **없음**으로 설정합니다.

이제 시트가 아래와 같이 보일 것입니다.

항공사 승객 분석을 위한 Qlik Sense 시트



데이터 해석 및 설명

STL 차트 함수를 사용하면 시계열 데이터에서 많은 정보를 얻을 수 있습니다.

추세 구성 요소

추세 구성 요소의 통계 정보가 비계절화됩니다. 이렇게 하면 시간이 지남에 따라 일반적이고 반복되지 않는 변동을 더 쉽게 볼 수 있습니다. *월 승객 수*에 대한 직선형 추세선과 비교할 때 STL 추세 구성 요소는 변화하는 추세를 캡처합니다. 여전히 읽기 쉬운 방식으로 정보를 제공하면서 일부 명확한 편차를 표시합니다. STL 알고리즘의 스무딩 동작은 이를 포착하는 데 도움이 되었습니다.

STL 추세 그래프에서 볼 수 있는 항공사 승객 수의 감소는 1950년대에 발생한 경기 침체의 경제적 영향의 일부로 설명될 수 있습니다.

계절성 구성 요소

추세가 제거된 계절성 구성 요소는 시계열 전체에서 반복되는 변동을 격리하고 분석의 해당 부분에서 일반적인 추세 정보를 제거했습니다. 연도-월 집계로 구성된 데이터 집합으로 시작했습니다. 이 데이터를 통해 데이터를 1개월 단위로 세분화하고 있음을 암시합니다. 기간 값을 12로 정의하여 1년(12개월) 주기 동안 계절성 패턴을 모델링하도록 차트를 설정했습니다.

데이터에는 여름에 항공사 승객이 급증한 후 겨울에 감소하는 반복적인 계절적 패턴이 있습니다. 이는 여름이 일반적으로 휴가와 여행을 떠나는 인기 있는 시간이라는 생각과 일치합니다. 또한 시계열이 진행되는 동안 이러한 계절성 주기의 진폭이 급격히 증가하는 것을 볼 수 있습니다.

잔차 구성 요소

잔차 구성 요소에 대한 차트에는 추세 및 계절성 분해에서 캡처되지 않은 모든 정보가 표시됩니다. 잔차 구성 요소에는 통계적 노이즈가 포함되지만 STL 추세 및 계절성 함수 인수의 잘못된 설정을 나타낼 수도 있습니다. 일반적으로 신호의 잔차 구성 요소에 주기적인 진동이 있거나 표시되는 정보가 분명히 임의적이지 않은 경우 일반적으로 현재 계절성 또는 추세 구성 요소에 캡처되지 않은 시계열 정보가 있다는 신호입니다. 이 경우 각 함수 인수의 정의를 다시 검토하고 가능하면 주기성을 변경해야 합니다.

스무더 값

추세 및 계절성 스무더에 대한 값을 지정하지 않았으므로 함수는 이러한 매개 변수에 대해 기본값을 사용합니다. Qlik Sense에서 STL 알고리즘의 기본 스무더 값은 효과적인 결과를 생성합니다. 결과적으로 대부분의 경우 이러한 인수를 표현식에서 제외할 수 있습니다.

세 가지 STL 함수 중 하나에서 계절성 또는 추세 스무더 인수를 0으로 설정하면 알고리즘이 0 값이 아닌 기본값을 사용합니다.

추세 스무더 값은 차트에 지정된 차원을 사용합니다. *YearMonth* 필드는 데이터를 월 단위로 표시하므로 추세 스무더 값은 월 수입니다. 계절성 스무더는 정의된 주기성을 반영합니다. 이 경우 한 기간을 12개월(1년)로 정의했으므로 계절성 스무더 값은 연 수입니다. 이는 혼란스럽게 들릴 수 있지만 실제로는 계절성을 찾으려면 여러 계절성을 살펴봐야 한다는 것을 의미합니다. 이 숫자는 계절성 스무더입니다.

기타 유용한 정보

계절성 주기가 시간이 지남에 따라 진폭이 증가한다는 점을 감안할 때 고급 분석 접근 방식은 대수 함수를 사용하여 곱셈 분해를 만들 수 있습니다. 실제로 계절성을 추세 구성 요소로 나누어 Qlik Sense에서 상대 진폭의 간단한 측정값을 만들 수 있습니다. 이 작업이 완료되면 시간이 지남에 따라 각 주기의 여름 피크가 상대적 진폭에서 더 커지는 것을 알 수 있습니다. 그러나 겨울철 저점의 진폭은 시간이 지남에 따라 증가하지 않습니다.

8.23 통계 분포 함수

통계 분포 함수는 주어진 입력 변수에 대해 다양한 가능한 결과가 발생할 확률을 반환합니다. 이러한 함수를 사용하여 데이터 포인트의 잠재적 값을 계산할 수 있습니다.

아래에 설명된 3개의 통계 분포 함수 그룹은 Qlik Sense에서 모두 Cephes 함수 라이브러리를 사용하여 구현됩니다. 사용된 알고리즘에 대한 참조 및 상세 정보 등은 Cephes library를 참조하십시오. Cephes 함수 라이브러리는 승인 하에 사용되었습니다.

- 확률 함수는 제공된 값에 의해 주어진 분포의 포인트에서 확률을 계산합니다.
  - 빈도 함수는 불연속 분포에 사용됩니다.
  - 밀도 함수는 연속 함수에 사용됩니다.

- Dist 함수는 제공된 값이 주어진 분포의 포인트에서 분포의 누적 확률을 계산합니다.
- Inv 함수는 분포의 누적 확률이 주어지면 역값을 계산합니다.

모든 함수는 데이터 로드 스크립트와 차트 표현식 모두에서 사용할 수 있습니다.

통계 분포 함수 개요

각 함수는 개요가 끝난 후에 더 자세히 설명합니다. 구문에서 함수 이름을 클릭하여 해당 함수에 대한 상세 설명에 즉시 액세스할 수도 있습니다.

BetaDensity
`BetaDensity()`는 베타 분포의 확률을 반환합니다.

```
BetaDensity (value, alpha, beta)
```

BetaDist
`BetaDist()`는 베타 분포의 누적 확률을 반환합니다.

```
BetaDist (value, alpha, beta)
```

BetaInv
`BetaINV()`는 베타 분포의 누적 확률의 역을 반환합니다.

```
BetaInv (prob, alpha, beta)
```

BinomDist
`BinomDist()`는 이항 분포의 누적 확률을 반환합니다.

```
BinomDist (value, trials, trial_probability)
```

BinomFrequency
`BinomFrequency()`는 이항 확률 분포를 반환합니다.

```
BinomFrequency (value, trials, trial_probability)
```

BinomInv
`BinomInv()`는 이항 분포의 누적 확률의 역을 반환합니다.

```
BinomInv (prob, trials, trial_probability)
```

ChiDensity
`ChiDensity()`는 chi^2 분포의 단측 확률을 반환합니다. chi^2 밀도 함수는 chi^2 테스트와 연관됩니다.

```
ChiDensity (value, degrees_freedom)
```

ChiDist
`ChiDist()`는 chi^2 분포의 단측 확률을 반환합니다. chi^2 분포는 chi^2 테스트와 연관됩니다.

```
ChiDist (value, degrees_freedom)
```

ChiInv
`ChiInv()`는 chi^2 분포의 단측 확률의 역을 반환합니다.

```
ChiInv (prob, degrees_freedom)
```

FDensity
FDensity()는 F 분포의 확률을 반환합니다.

```
FDensity (value, degrees_freedom1, degrees_freedom2)
```

FDist
FDist()는 F 분포의 누적 확률을 반환합니다.

```
FDist (value, degrees_freedom1, degrees_freedom2)
```

FInv
FInv()는 F 분포의 누적 확률의 역을 반환합니다.

```
FInv (prob, degrees_freedom1, degrees_freedom2)
```

GammaDensity
GammaDensity()는 감마 분포의 확률을 반환합니다.

```
GammaDensity (value, k, θ)
```

GammaDist
GammaDist()는 감마 분포의 누적 확률을 반환합니다.

```
GammaDist (value, k, θ)
```

GammaInv
GammaInv()는 감마 분포의 누적 확률의 역을 반환합니다.

```
GammaInv (prob, k, θ)
```

NormDist
NormDist()는 지정된 평균 및 표준 편차에 대한 누적 정규 분포를 반환합니다. mean = 0이고 standard_dev = 1인 경우 이 함수는 표준 정규 분포를 반환합니다.

```
NormDist (value, mean, standard_dev)
```

NormInv
NormInv()는 지정된 평균 및 표준 편차에 대한 정규 누적 분포의 역을 반환합니다.

```
NormInv (prob, mean, standard_dev)
```

PoissonDist
PoissonDist()는 푸아송 분포의 누적 확률을 반환합니다.

```
PoissonDist (value, mean)
```

PoissonFrequency
PoissonFrequency()는 푸아송 확률 분포를 반환합니다.

```
PoissonFrequency (value, mean)
```

PoissonInv

PoissonInv()는 푸아송 분포의 누적 확률의 역을 반환합니다.

```
PoissonInv (prob, mean)
```

TDensity

TDensity() 는 학생 t 밀도 함수에 대한 값을 반환합니다. 여기서 숫자 값은 확률을 계산할 t에 대해 계산된 값입니다.

```
TDensity (value, degrees_freedom, tails)
```

TDist

TDist()는 학생 t 분포에 대한 확률을 반환합니다. 여기서 숫자 값은 확률을 계산할 t에 대해 계산된 값입니다.

```
TDist (value, degrees_freedom, tails)
```

TInv

TInv()는 확률 및 자유도의 함수로서 학생 t 분포의 t 값을 반환합니다.

```
TInv (prob, degrees_freedom)
```

관련 항목:

통계 집계 함수 (page 383)

BetaDensity

BetaDensity()는 베타 분포의 확률을 반환합니다.

구문:

```
BetaDensity(value, alpha, beta)
```

반환 데이터 유형: 숫자

인수

인수	설명
value	분포를 평가할 값입니다. 이 값은 0과 1 사이여야 합니다.
alpha	첫 번째 형상 매개 변수를 정의하는 양수입니다. 임의 변수의 지수입니다.
beta	두 번째 형상 매개 변수를 정의하는 양수입니다. 분모 자유도의 수를 나타냅니다.

BetaDist

BetaDist()는 베타 분포의 누적 확률을 반환합니다.

구문:

```
BetaDist(value, alpha, beta)
```

반환 데이터 유형: 숫자

인수

인수	설명
value	분포를 평가할 값입니다. 이 값은 0과 1 사이여야 합니다.
alpha	첫 번째 형상 매개 변수를 정의하는 양수입니다. 임의 변수의 지수입니다.
beta	두 번째 형상 매개 변수를 정의하는 양수입니다. 이는 분포의 모양을 제어하는 지수입니다.

이 함수는 다음과 같은 방식으로 BetaInv 함수와 연관됩니다.
If prob = BetaDist(value, alpha, beta), then BetaInv(prob, alpha, beta) = value

BetaInv

BetaINV()는 베타 분포의 누적 확률의 역을 반환합니다.

구문:

```
BetaInv(prob, alpha, beta)
```

반환 데이터 유형: 숫자

인수

인수	설명
prob	베타 확률 분포와 관련된 확률입니다. 0 ~ 1 사이의 숫자여야 합니다.
alpha	첫 번째 형상 매개 변수를 정의하는 양수입니다. 임의 변수의 지수입니다.
beta	두 번째 형상 매개 변수를 정의하는 양수입니다. 이는 분포의 모양을 제어하는 지수입니다.

이 함수는 다음과 같은 방식으로 BetaDist 함수와 연관됩니다.
If prob = BetaDist(value, alpha, beta), then BetaInv(prob, alpha, beta) = value

BinomDist

BinomDist()는 이항 분포의 누적 확률을 반환합니다.

구문:

```
BinomDist(value, trials, trial_probability)
```

반환 데이터 유형: 숫자

인수

인수	설명
value	분포를 평가할 값입니다. 이 값은 0보다 작지 않고 시도 횟수보다 크지 않은 정수여야 합니다.

인수	설명
trials	시도 횟수를 나타내는 양의 정수입니다.
trial_probability	각 시도의 성공 확률입니다. 항상 0과 1 사이의 숫자입니다.

이 함수는 다음과 같은 방식으로 BinomInv 함수와 연관됩니다.
If prob = BinomDIST(value, trials, trial_probability), then BinomInv(prob, trials, trial_probability) = value

BinomFrequency

BinomFrequency()는 이항 확률 분포를 반환합니다.

구문:

```
BinomFrequency(value, trials, trial_probability)
```

반환 데이터 유형: 숫자

인수

인수	설명
value	분포를 평가할 값입니다. 이 값은 0보다 작지 않고 시도 횟수보다 크지 않은 정수여야 합니다.
trials	시도 횟수를 나타내는 양의 정수입니다.
trial_probability	각 시도의 성공 확률입니다. 항상 0과 1 사이의 숫자입니다.

BinomInv

BinomInv()는 이항 분포의 누적 확률의 역을 반환합니다.

구문:

```
BinomInv(prob, trials, trial_probability)
```

반환 데이터 유형: 숫자

인수

인수	설명
prob	이항 확률 분포와 관련된 확률입니다. 0 ~ 1 사이의 숫자여야 합니다.
trials	시도 횟수를 나타내는 양의 정수입니다.
trial_probability	각 시도의 성공 확률입니다. 항상 0과 1 사이의 숫자입니다.

이 함수는 다음과 같은 방식으로 BinomDist 함수와 연관됩니다.
If prob = BinomDist(value, trials, trial_probability), then BinomInv(prob, trials, trial_probability) = value

ChiDensity

ChiDensity()는 chi^2 분포의 단측 확률을 반환합니다. chi^2 밀도 함수는 chi^2 테스트와 연관됩니다.

구문:

```
ChiDensity(value, degrees_freedom)
```

반환 데이터 유형: 숫자

인수

인수	설명
value	분포를 평가할 값입니다. 값은 음수가 아니어야 합니다.
degrees_freedom	분자 자유도 수를 나타내는 양의 정수입니다.

ChiDist

ChiDist()는 chi^2 분포의 단측 확률을 반환합니다. chi^2 분포는 chi^2 테스트와 연관됩니다.

구문:

```
CHIDIST(value, degrees_freedom)
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
value	분포를 평가할 값입니다. 값은 음수가 아니어야 합니다.
degrees_freedom	자유도 수를 나타내는 양의 정수입니다.

이 함수는 다음과 같은 방식으로 **ChiInv** 함수와 연관됩니다.
If prob = CHIDIST(value,df), then CHIINV(prob, df) = value

제한 사항:

모든 인수는 숫자여야 하며, 그렇지 않으면 NULL이 반환됩니다.

예 및 결과:

예	결과
CHIDIST(8, 15)	0.9238을 반환합니다.

ChiInv

ChiInv()는 chi^2 분포의 단측 확률의 역을 반환합니다.

구문:

```
CHIINV(prob, degrees_freedom)
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
prob	chi^2 분포와 관련된 확률이며, 0 ~ 1 사이의 숫자여야 합니다.
degrees_freedom	자유도 수를 나타내는 정수입니다.

이 함수는 다음과 같은 방식으로 **ChiDist** 함수와 연관됩니다.
`If prob = CHIDIST(value,df), then CHIINV(prob, df) = value`

제한 사항:

모든 인수는 숫자여야 하며, 그렇지 않으면 NULL이 반환됩니다.

예 및 결과:

예	결과
CHIINV(0.9237827, 15)	8.0000을 반환합니다.

FDensity

FDensity()는 F 분포의 확률을 반환합니다.

구문:

```
FDensity(value, degrees_freedom1, degrees_freedom2)
```

반환 데이터 유형: 숫자

인수

인수	설명
value	분포를 평가할 값입니다. 값은 음수가 아니어야 합니다.
degrees_freedom1	분자 자유도 수를 나타내는 양의 정수입니다.
degrees_freedom2	분모 자유도 수를 나타내는 양의 정수입니다.

FDist

FDist()는 F 분포의 누적 확률을 반환합니다.

구문:

```
FDist(value, degrees_freedom1, degrees_freedom2)
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
value	분포를 평가할 값입니다. 값은 음수가 아니어야 합니다.
degrees_freedom1	분자 자유도 수를 나타내는 양의 정수입니다.
degrees_freedom2	분모 자유도 수를 나타내는 양의 정수입니다.

이 함수는 다음과 같은 방식으로 **FInv** 함수와 연관됩니다.
`If prob = FDIST(value, df1, df2), then FINV(prob, df1, df2) = value`

제한 사항:

모든 인수는 숫자여야 하며, 그렇지 않으면 NULL이 반환됩니다.

예 및 결과:

예	결과
FDIST(15, 8, 6)	0.0019를 반환합니다.

FInv

FInv()는 F 분포의 누적 확률의 역을 반환합니다.

구문:

```
FInv(prob, degrees_freedom1, degrees_freedom2)
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
prob	F 확률 분포와 관련된 확률이며, 0 ~ 1 사이의 숫자여야 합니다.
degrees_freedom	자유도 수를 나타내는 정수입니다.

이 함수는 다음과 같은 방식으로 **FDist** 함수와 연관됩니다.
`If prob = FDIST(value, df1, df2), then FINV(prob, df1, df2) = value`

제한 사항:

모든 인수는 숫자여야 하며, 그렇지 않으면 NULL이 반환됩니다.

예 및 결과:

예	결과
FINV(0.0019369, 8, 6)	15.0000을 반환합니다.

GammaDensity

`GammaDensity()`는 감마 분포의 확률을 반환합니다.

구문:

```
GammaDensity(value, k, θ)
```

반환 데이터 유형: 숫자

인수

인수	설명
value	분포를 평가할 값입니다. 값은 음수가 아니어야 합니다.
k	형상 매개 변수를 정의하는 양수입니다.
θ	척도 매개 변수를 정의하는 양수입니다.

GammaDist

`GammaDist()`는 감마 분포의 누적 확률을 반환합니다.

구문:

```
GammaDist(value, k, θ)
```

반환 데이터 유형: 숫자

인수

인수	설명
value	분포를 평가할 값입니다. 값은 음수가 아니어야 합니다.
k	형상 매개 변수를 정의하는 양수입니다.
θ	척도 매개 변수를 정의하는 양수입니다.

이 함수는 다음과 같은 방식으로 `GammaINV` 함수와 연관됩니다.
`If prob = GammaDist(value, k, θ), then GammaInv(prob, k, θ) = value`

GammaInv

GammaInv()는 감마 분포의 누적 확률의 역을 반환합니다.

구문:

```
GammaInv(prob, k, θ)
```

반환 데이터 유형: 숫자

인수

인수	설명
prob	감마 확률 분포와 관련된 확률입니다. 0 ~ 1 사이의 숫자여야 합니다.
k	형상 매개 변수를 정의하는 양수입니다.
θ	척도 매개 변수를 정의하는 양수입니다.

이 함수는 다음과 같은 방식으로 GammaDist 함수와 연관됩니다.
If prob = GammaDist(value, k, θ), then GammaInv(prob, k, θ) = value

NormDist

NormDist()는 지정된 평균 및 표준 편차에 대한 누적 정규 분포를 반환합니다. mean = 0이고 standard_dev = 1인 경우 이 함수는 표준 정규 분포를 반환합니다.

구문:

```
NORMDIST(value, [mean], [standard_dev], [cumulative])
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
value	분포를 평가할 값입니다.
mean	분포의 산술 평균을 나타내는 선택적 값입니다. 이 인수를 지정하지 않는 경우 기본값은 0입니다.
standard_dev	분포의 표준 편차를 나타내는 선택적 양수 값입니다. 이 인수를 지정하지 않는 경우 기본값은 1입니다.
cumulative	필요에 따라 표준 정규 분포 또는 누적 분포를 사용하도록 선택할 수 있습니다. 0 = 표준 정규 분포 1 = 누적 분포(기본값)

이 함수는 다음과 같은 방식으로 **NormInv** 함수와 연관됩니다.
`If prob = NORMDIST(value, m, sd), then NORMINV(prob, m, sd) = value`

제한 사항:

모든 인수는 숫자여야 하며, 그렇지 않으면 NULL이 반환됩니다.

예 및 결과:

예	결과
NORMDIST(0.5, 0, 1)	0.6915를 반환합니다.

NormInv

`NormInv()`는 지정된 평균 및 표준 편차에 대한 정규 누적 분포의 역을 반환합니다.

구문:

```
NORMINV(prob, mean, standard_dev)
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
prob	정규 분포와 관련된 확률이며, 0 ~ 1 사이의 숫자여야 합니다.
mean	분포의 산술 평균을 나타내는 값입니다.
standard_dev	분포의 표준 편차를 나타내는 양수 값입니다.

이 함수는 다음과 같은 방식으로 **NormDist** 함수와 연관됩니다.
`If prob = NORMDIST(value, m, sd), then NORMINV(prob, m, sd) = value`

제한 사항:

모든 인수는 숫자여야 하며, 그렇지 않으면 NULL이 반환됩니다.

예 및 결과:

예	결과
NORMINV(0.6914625, 0, 1)	0.5000을 반환합니다.

PoissonDist

`PoissonDist()`는 푸아송 분포의 누적 확률을 반환합니다.

구문:

```
PoissonDist(value, mean)
```

반환 데이터 유형: 숫자

인수

인수	설명
value	분포를 평가할 값입니다. 값은 음수가 아니어야 합니다.
mean	평균 결과를 정의하는 양수입니다.

이 함수는 다음과 같은 방식으로 PoissonInv 함수와 연관됩니다.
If prob = PoissonDist(value, mean), then PoissonInv(prob, mean) = value

PoissonFrequency

PoissonFrequency()는 푸아송 확률 분포를 반환합니다.

구문:

```
PoissonFrequency(value, mean)
```

반환 데이터 유형: 숫자

인수

인수	설명
value	분포를 평가할 값입니다. 값은 음수가 아니어야 합니다.
mean	평균 결과를 정의하는 양수입니다.

PoissonInv

PoissonInv()는 푸아송 분포의 누적 확률의 역을 반환합니다.

구문:

```
PoissonInv(prob, mean)
```

반환 데이터 유형: 숫자

인수

인수	설명
prob	푸아송 확률 분포와 관련된 확률입니다. 0 ~ 1 사이의 숫자여야 합니다.
mean	평균 결과를 정의하는 양수입니다.

이 함수는 다음과 같은 방식으로 PoissonDIST 함수와 연관됩니다.
If prob = PoissonDist(value, mean), then PoissonInv(prob, mean) = value

TDensity

TDensity() 는 학생 t 밀도 함수에 대한 값을 반환합니다. 여기서 숫자 값은 확률을 계산할 t에 대해 계산된 값입니다.

구문:

```
TDensity(value, degrees_freedom)
```

반환 데이터 유형: 숫자

인수

인수	설명
value	분포를 평가할 값입니다. 값은 음수가 아니어야 합니다.
degrees_freedom	자유도 수를 나타내는 양의 정수입니다.

TDist

TDist()는 학생 t 분포에 대한 확률을 반환합니다. 여기서 숫자 값은 확률을 계산할 t에 대해 계산된 값입니다.

구문:

```
TDist(value, degrees_freedom, tails)
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
value	분포를 평가할 값입니다. 값은 음수가 아니어야 합니다.
degrees_freedom	자유도 수를 나타내는 양의 정수입니다.
tails	1(단측 분포) 또는 2(양측 분포) 중 하나여야 합니다.

이 함수는 다음과 같은 방식으로 **TInv** 함수와 연관됩니다.
`If prob = TDIST(value, df ,2), then TINV(prob, df) = value`

제한 사항:

모든 인수는 숫자여야 하며, 그렇지 않으면 NULL이 반환됩니다.

예 및 결과:

예	결과
TDIST(1, 30, 2)	0.3253을 반환합니다.

TInv

TInv()는 확률 및 자유도의 함수로서 학생 t 분포의 t 값을 반환합니다.

구문:

```
TINV(prob, degrees_freedom)
```

반환 데이터 유형: 숫자

인수:

인수

인수	설명
prob	t 분포와 관련된 양측 확률이며, 0 ~ 1 사이의 숫자여야 합니다.
degrees_freedom	자유도 수를 나타내는 정수입니다.

제한 사항:

모든 인수는 숫자여야 하며, 그렇지 않으면 NULL이 반환됩니다.

이 함수는 다음과 같은 방식으로 **TDist** 함수와 연관됩니다.
`If prob = TDIST(value, df ,2), then TINV(prob, df) = value.`

예 및 결과:

예	결과
TINV(0.3253086, 30)	1.0000을 반환합니다.

8.24 문자열 함수

이 섹션에서는 문자열을 처리하는 함수에 대해 설명합니다.

데이터 로드 스크립트에서만 사용할 수 있는 **Evaluate**를 제외한 모든 함수는 데이터 로드 스크립트와 차트 표현식 모두에서 사용할 수 있습니다.

문자열 함수 개요

각 함수는 개요가 끝난 후에 더 자세히 설명합니다. 구문에서 함수 이름을 클릭하여 해당 함수에 대한 상세 설명에 즉시 액세스할 수도 있습니다.

Capitalize
Capitalize()는 모든 단어의 첫 글자가 대문자인 문자열을 반환합니다.

```
Capitalize (text)
```

Chr
Chr()은 입력된 정수에 해당하는 Unicode 문자를 반환합니다.

```
Chr (int)
```

Evaluate

Evaluate()는 입력 텍스트 문자열을 유효한 Qlik Sense 표현식으로 평가할 수 있는지 확인하고, 평가할 수 있는 경우는 표현식의 값을 문자열로 반환합니다. 입력 문자열이 유효한 표현식이 아니면 NULL이 반환됩니다.

```
Evaluate (expression_text)
```

FindOneOf

FindOneOf()는 제공된 문자 집합의 문자 발견 위치를 찾기 위해 문자열을 검색합니다. 세 번째 인수(1보다 큰 값으로)를 지정하지 않으면 검색 집합의 문자에 대한 첫 번째 발견 위치가 반환됩니다. 일치 항목이 발견되지 않으면 **0**이 반환됩니다.

```
FindOneOf (text, char_set[, count])
```

Hash128

Hash128()은 결합된 입력 표현식 값의 128비트 해시를 반환합니다. 결과는 22자의 문자열입니다.

```
Hash128 (expr{, expression})
```

Hash160

Hash160()은 결합된 입력 표현식 값의 160비트 해시를 반환합니다. 결과는 27자의 문자열입니다.

```
Hash160 (expr{, expression})
```

Hash256

Hash256()은 결합된 입력 표현식 값의 256비트 해시를 반환합니다. 결과는 43자의 문자열입니다.

```
Hash256 (expr{, expression})
```

Index

Index()는 제공된 부분 문자열의 n번째 항목의 시작 위치를 찾기 위해 문자열을 검색합니다. 세 번째 선택적 인수가 n 값을 제공하며, 생략 시 1입니다. 음수 값은 문자열 끝부터 검색합니다. 문자열 내 위치는 **1**부터 번호가 매겨집니다.

```
Index  (text, substring[, count])
```

IsJson

IsJson()은 지정된 문자열에 유효한 JSON(JavaScript Object Notation) 데이터가 포함되어 있는지 테스트합니다. 특정 JSON 데이터 유형의 유효성을 검사할 수도 있습니다.

```
IsJson (json [, type])
```

JsonGet

JsonGet()은 JSON(JavaScript Object Notation) 데이터 문자열의 경로를 반환합니다. 데이터는 유효한 JSON이어야 하지만 추가 공백이나 줄 바꿈을 포함할 수 있습니다.

```
JsonGet (json, path)
```

JsonSet

JsonSet()은 JSON(JavaScript Object Notation) 데이터가 포함된 문자열을 수정합니다. 경로로 지정된 새 위치로 JSON 값을 설정하거나 삽입할 수 있습니다. 데이터는 유효한 JSON이어야 하지만 추가 공백이나 줄 바꿈을 포함할 수 있습니다.

```
JsonSet(json, path, value)
```

KeepChar

KeepChar()은 첫 번째 문자열 'text'로 이루어진 문자열 중 두 번째 문자열 "keep_chars"에 포함되지 않는 문자를 뺀 문자열을 반환합니다.

```
KeepChar (text, keep_chars)
```

Left

Left()는 입력 문자열의 첫 번째(가장 왼쪽) 문자로 구성된 문자열을 반환하며 문자 수는 두 번째 인수로 결정됩니다.

```
Left (text, count)
```

Len

Len()은 입력 문자열의 길이를 반환합니다.

```
Len (text)
```

LevenshteinDist

LevenshteinDist()는 두 문자열 사이의 Levenshtein 거리를 반환합니다. 이는 한 문자열을 다른 문자열로 변경하는 데 필요한 최소 단일 문자 편집 수(삽입, 삭제 또는 대체)로 정의됩니다. 이 함수는 퍼지 문자열 비교에 유용합니다.

```
LevenshteinDist (text1, text2)
```

Lower

Lower()는 입력 문자열의 모든 문자를 소문자로 변환합니다.

```
Lower (text)
```

LTrim

LTrim()은 모든 선행 공백이 제거된 입력 문자열을 반환합니다.

```
LTrim (text)
```

Mid

Mid()는 두 번째 인수 'start'로 정의된 문자의 위치에서 시작되고 세 번째 인수 'count'로 정의된 문자 수를 반환하는 입력 문자열의 일부를 반환합니다. 'count'를 생략하면 입력 문자열의 나머지가 반환됩니다. 입력 문자열의 첫 번째 문자는 번호가 1로 지정됩니다.

```
Mid (text, start[, count])
```

Ord

Ord()는 입력 문자열의 첫 번째 문자에 대한 Unicode 코드 포인트 번호를 반환합니다.

```
Ord (text)
```

PurgeChar

PurgeChar()은 두 번째 인수('remove_chars')에 나타나는 문자를 제외하고, 입력 문자열('text')에 포함된 문자로 구성된 문자열을 반환합니다.

```
PurgeChar (text, remove_chars)
```

Repeat

Repeat()는 두 번째 인수로 정의된 횟수 만큼 반복된 입력 문자열로 구성된 문자열을 만듭니다.

```
Repeat (text[, repeat_count])
```

Replace

Replace()는 입력 문자열 내에서 특정 부분 문자열의 모든 발견 항목을 다른 부분 문자열로 대체한 후의 문자열을 반환합니다. 이 함수는 비재귀적이며 왼쪽에서 오른쪽으로 작동합니다.

```
Replace (text, from_str, to_str)
```

Right

Right()는 입력 문자열의 마지막(가장 오른쪽) 문자로 구성된 문자열을 반환합니다. 문자 수는 두 번째 인수로 결정됩니다.

```
Right (text, count)
```

RTrim

RTrim()은 모든 후행 공백이 제거된 입력 문자열을 반환합니다.

```
RTrim (text)
```

SubField

SubField()는 부모 문자열 필드에서 부분 문자열 성분을 추출하는 데 사용됩니다. 여기서 원래 레코드 필드는 구분 기호로 분리된 둘 이상의 부분으로 구성됩니다.

```
SubField (text, delimiter[, field_no ])
```

SubStringCount

SubStringCount()는 입력 문자열 텍스트에 지정된 부분 문자열의 발생 횟수를 반환합니다. 일치하는 항목이 없으면 0이 반환됩니다.

```
SubStringCount (text, substring)
```

TextBetween

TextBetween()은 구분 기호로 지정된 문자 사이에서 발견되는 입력 문자열의 텍스트를 반환합니다.

```
TextBetween (text, delimiter1, delimiter2[, n])
```

Trim

Trim()은 모든 선행 및 후행 공백이 제거된 입력 문자열을 반환합니다.

```
Trim (text)
```

Upper

Upper()는 표현식의 모든 텍스트 문자에 대해 입력 문자열의 모든 문자를 대문자로 변환합니다. 숫자와 기호는 무시됩니다.

```
Upper (text)
```

Capitalize

Capitalize()는 모든 단어의 첫 글자가 대문자인 문자열을 반환합니다.

구문:

```
Capitalize(text)
```

반환 데이터 유형: 문자열

예: 차트 표현식

예	결과
`Capitalize ( 'star trek' )`	'Star Trek'를 반환합니다.
`Capitalize ( 'AA bb cC Dd')`	'Aa Bb Cc Dd'를 반환합니다.

예: 로드 스크립트

```
Load  String, Capitalize(String) Inline [String rHode iSland washingTon d.C. new york];
```

결과

문자열	Capitalize(String)
rHode iSland	Rhode Island
washingTon d.C.	Washington D.C.
new york	New York

Chr

Chr()은 입력된 정수에 해당하는 Unicode 문자를 반환합니다.

구문:

```
Chr(int)
```

반환 데이터 유형: 문자열

예 및 결과:

예	결과
Chr(65)	문자열 'A'를 반환합니다.

예	결과
Chr(163)	문자열 '£'를 반환합니다.
Chr(35)	문자열 '#'를 반환합니다.

Evaluate

Evaluate()는 입력 텍스트 문자열을 유효한 Qlik Sense 표현식으로 평가할 수 있는지 확인하고, 평가할 수 있는 경우는 표현식의 값을 문자열로 반환합니다. 입력 문자열이 유효한 표현식이 아니면 NULL이 반환됩니다.

구문:

```
Evaluate(expression_text)
```

반환 데이터 유형: dual

이 문자열 함수는 차트 표현식에 사용할 수 없습니다.

예 및 결과:

함수 예	결과
`Evaluate ( 5 * 8 )`	'40'를 반환합니다.

로드 스크립트 예

```
Load  Evaluate(String) as Evaluated, String Inline [String 4 5+3 0123456789012345678 Today()
];
```

결과

문자열	평가
4	4
5+3	8
0123456789012345678	0123456789012345678
Today()	2022-02-02

FindOneOf

FindOneOf()는 제공된 문자 집합의 문자 발견 위치를 찾기 위해 문자열을 검색합니다. 세 번째 인수(1보다 큰 값으로)를 지정하지 않으면 검색 집합의 문자에 대한 첫 번째 발견 위치가 반환됩니다. 일치 항목이 발견되지 않으면 **0**이 반환됩니다.

구문:

```
FindOneOf(text, char_set[, count])
```

반환 데이터 유형: 정수

인수:

인수

인수	설명
text	원래 문자열입니다.
char_set	text에서 검색할 일련의 문자들입니다.
count	검색할 문자열의 발생 위치를 정의합니다. 예를 들어 값이 2이면 두 번째 발생 항목을 검색합니다.

예: 차트 표현식

예	결과
`FindOneOf( 'my example text string', 'et%s')`	'e'가 문자열 예의 네 번째 문자이기 때문에 '4'를 반환합니다.
`FindOneOf( 'my example text string', 'et%s', 3)`	검색이 문자 e, t, % 또는 s 중 하나를 검색하고 "t"가 문자열 예의 위치 12에서 세 번째 발생 항목이므로 '12'를 반환합니다.
`FindOneOf( 'my example text string', '¤%&')`	문자열 예에 문자 ¤, % 또는 &가 없으므로 '0'을 반환합니다.

예: 로드 스크립트

```
Load * Inline [SearchFor, Occurrence et%s,1 et%s,3 ¤%&,1]
```

결과

SearchFor	Occurrence	FindOneOf('my example text string', SearchFor, Occurrence)
et%s	1	4
et%s	3	12
¤%&	1	0

Hash128

Hash128()은 결합된 입력 표현식 값의 128비트 해시를 반환합니다. 결과는 22자의 문자열입니다.

구문:

```
Hash128(expr{, expression})
```

반환 데이터 유형: 문자열

예: 차트 표현식

예	결과
Hash128 ('abc', 'xyz', '123')	'MA&5]6+3=:>:>G%S<U*S2+'를 반환합니다.
Hash128 (Region, Year, Month) Note: Region, Year, and Month are table fields.	'G7*=6GKPJ(Z+)^KM?<$'A+'를 반환합니다.

예: 로드 스크립트

```
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 ];
```

결과

지역	연도	월	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()은 결합된 입력 표현식 값의 160비트 해시를 반환합니다. 결과는 27자의 문자열입니다.

구문:

```
Hash160(expr{, expression})
```

반환 데이터 유형: 문자열

예: 차트 표현식

예	결과
Hash160 ('abc', 'xyz', '123')	'MA&5]6+3=:>;>G%S<U*S2I:`=X*'를 반환합니다.
Hash160 (Region, Year, Month) Note: Region, Year, and Month are table fields.	'G7*=6GKPJ(Z+)^KM?<$'AI.)?U$'를 반환합니다.

예: 로드 스크립트

```
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 ];
```

결과

지역	연도	월	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()은 결합된 입력 표현식 값의 256비트 해시를 반환합니다. 결과는 43자의 문자열입니다.

구문:

```
Hash256(expr{, expression})
```

반환 데이터 유형: 문자열

예: 차트 표현식

예	결과
Hash256 ('abc', 'xyz', '123')	'MA&5]6+3=:>;>G%S<U*S2I:`=X*A.IO*8N\%Y7Q;YEJ'를 반환합니다.
Hash256 (Region, Year, Month) Note: Region, Year, and Month are table fields.	'G7*=6GKPJ(Z+)^KM?<$'AI.)?U$#X2RB[:0ZP=+Z`F:'을 반환합니다.

예: 로드 스크립트

```
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 ];
```

결과

지역	연도	월	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()는 제공된 부분 문자열의 n번째 항목의 시작 위치를 찾기 위해 문자열을 검색합니다. 세 번째 선택적 인수가 n 값을 제공하며, 생략 시 1입니다. 음수 값은 문자열 끝부터 검색합니다. 문자열 내 위치는 **1**부터 번호가 매겨집니다.

구문:

```
Index(text, substring[, count])
```

반환 데이터 유형: 정수

인수:

인수

인수	설명
text	원래 문자열입니다.
substring	text에서 검색할 문자열입니다.
count	검색할 **substring**의 발생 위치를 정의합니다. 예를 들어 값이 2이면 두 번째 발생 항목을 검색합니다.

예 및 결과:

예	결과
Index('abcdefg', 'cd')	3을 반환합니다.
Index('abcdabcd', 'b', 2)	6('b'의 두 번째 발생 위치)을 반환합니다.
Index('abcdabcd', 'b',-2)	2(끝에서부터 'b'의 두 번째 발생 위치)를 반환합니다.
`Left( Date, Index( Date,'-') -1 )` where **Date** = 1997-07-14	1997을 반환합니다.
`Mid( Date, Index( Date, '-', 2 ) -2, 2 )` where **Date** = 1997-07-14	07을 반환합니다.

스크립트

```
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()은 지정된 문자열에 유효한 JSON(JavaScript Object Notation) 데이터가 포함되어 있는지 테스트합니다. 특정 JSON 데이터 유형의 유효성을 검사할 수도 있습니다.

구문:

```
value IsJson(json [, type])
```

반환 데이터 유형: dual

인수

인수	설명
json	테스트할 문자열입니다. 추가 공백이나 줄 바꿈을 포함할 수 있습니다.
type	테스트할 JSON 데이터 유형을 지정하는 선택적 인수입니다. • 'value'(기본값) • 'object' • 'array' • 'string' • 'number' • 'Boolean' • 'null'

예: 유효한 JSON 및 유형

예	결과
`IsJson('null')`	-1 (true)를 반환합니다.
`IsJson('"abc"', 'value')`	-1 (true)를 반환합니다.
`IsJson('"abc"', 'string')`	-1 (true)를 반환합니다.
`IsJson(123, 'number')`	-1 (true)를 반환합니다.

예: 유효하지 않은 JSON 또는 유형

예	결과	설명
`IsJson('text')`	0 (false)를 반환합니다.	'text'는 유효한 JSON 값이 아닙니다
`IsJson('"text"', 'number')`	0 (false)를 반환합니다.	'"text"'는 유효한 JSON 숫자가 아닙니다
`IsJson('"text"', 'text')`	0 (false)를 반환합니다.	'text'는 유효한 JSON 유형이 아닙니다

JsonGet

JsonGet()은 JSON(JavaScript Object Notation) 데이터 문자열의 경로를 반환합니다. 데이터는 유효한 JSON이어야 하지만 추가 공백이나 줄 바꿈을 포함할 수 있습니다.

구문:

```
value JsonGet(json, path)
```

반환 데이터 유형: dual

인수

인수	설명
json	JSON 데이터를 포함하는 문자열입니다.
path	경로는 RFC 6901에 따라 지정해야 합니다. 이렇게 하면 복잡한 하위 문자열이나 인덱스 함수를 사용하지 않고도 JSON 데이터 내부의 속성을 조회할 수 있습니다.

예: 유효한 JSON 및 경로

예	결과
`JsonGet('{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}', '')`	'{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}'를 반환합니다.
`JsonGet('{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}', '/a')`	'{"foo":"bar"}'를 반환합니다.
`JsonGet('{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}', '/a/foo')`	'"bar"'를 반환합니다.
`JsonGet('{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}', '/b')`	'[123,"abc","ABC"]'를 반환합니다.
`JsonGet('{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}', '/b/0')`	'123'를 반환합니다.
`JsonGet('{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}', '/b/1')`	'"abc"'를 반환합니다.
`JsonGet('{"a":{"foo":"bar"},"b":[123,"abc","ABC"]}', '/b/2')`	'"ABC"'를 반환합니다.

예: 유효하지 않은 JSON 또는 경로

예	결과	설명
`JsonGet('{"a":"b"}','/b')`	null를 반환합니다.	경로가 JSON 데이터의 유효한 부분을 가리키지 않습니다.
`JsonGet('{"a"}','/a')`	null를 반환합니다.	JSON 데이터가 유효한 JSON이 아닙니다(구성원 "a"에 값이 없음).

JsonSet

JsonSet()은 JSON(JavaScript Object Notation) 데이터가 포함된 문자열을 수정합니다. 경로로 지정된 새 위치로 JSON 값을 설정하거나 삽입할 수 있습니다. 데이터는 유효한 JSON이어야 하지만 추가 공백이나 줄 바꿈을 포함할 수 있습니다.

구문:

```
value JsonSet(json, path, value)
```

반환 데이터 유형: dual

인수

인수	설명
json	JSON 데이터를 포함하는 문자열입니다.
path	경로는 RFC 6901에 따라 지정해야 합니다. 이를 통해 복잡한 하위 문자열 또는 인덱스 함수 및 연결을 사용하지 않고도 JSON 데이터 내부에 속성을 구축할 수 있습니다.
value	JSON 형식의 새 문자열 값입니다.

예: 유효한 JSON, 경로 및 값

예	결과
`JsonSet('{}','/a','"b"')`	'{"a":"b"}'를 반환합니다.
`JsonSet('[]','/0','"x"')`	'["x"]'를 반환합니다.
`JsonSet('"abc"','','123')`	123를 반환합니다.

예: 유효하지 않은 JSON, 경로 또는 값

예	결과	설명
`JsonSet('"abc"','/x','123')`	null를 반환합니다.	경로가 JSON 데이터의 유효한 부분을 가리키지 않습니다.
`JsonSet('{"a":{"b":"c"}}','a/b','"x"')`	null를 반환합니다.	경로가 잘못되었습니다.
`JsonSet('{"a":"b"}','/a','abc')`	null를 반환합니다.	값이 유효한 JSON이 아닙니다. 문자열은 따옴표로 묶어야 합니다.

KeepChar

KeepChar()은 첫 번째 문자열 'text'로 이루어진 문자열 중 두 번째 문자열 "keep_chars"에 포함되지 않는 문자를 뺀 문자열을 반환합니다.

구문:

```
KeepChar(text, keep_chars)
```

반환 데이터 유형: 문자열

인수:

인수

인수	설명
text	원래 문자열입니다.
keep_chars	유지할 text의 문자가 포함된 문자열입니다.

예: 차트 표현식

예	결과
`KeepChar ( 'a1b2c3','123' )`	'123'을 반환합니다.
`KeepChar ( 'a1b2c3','1234' )`	'123'을 반환합니다.
`KeepChar ( 'a1b22c3','1234' )`	'1223'을 반환합니다.
`KeepChar ( 'a1b2c3','312' )`	'123'을 반환합니다.

예: 로드 스크립트

```
T1:
Load
*,
keepchar(String1, String2) as KeepChar;
Load * inline [
String1, String2
'a1b2c3', '123'
];
```

결과

로드 스크립트에서 *KeepChar* 함수를 사용하여 나온 출력을 보여 주는 Qlik Sense 테이블

String1	String2	KeepChar
a1b2c3	123	123

관련 항목:

PurgeChar (page 1411)

Left

Left()는 입력 문자열의 첫 번째(가장 왼쪽) 문자로 구성된 문자열을 반환하며 문자 수는 두 번째 인수로 결정됩니다.

구문:

```
Left(text, count)
```

반환 데이터 유형: 문자열

인수:

인수	설명
text	원래 문자열입니다.
count	문자열 **text**의 맨 왼쪽 부분부터 포함시킬 문자 수를 정의합니다.

예: 차트 표현식

예	결과
`Left('abcdef', 3)`	'abc'를 반환합니다.

예: 로드 스크립트

```
T1: Load  *,  left(Text,Start) as Left;        Load * inline [ Text, Start 'abcdef', 3 '2021-07-14', 4 '2021-07-14', 2 ];
```

결과

로드 스크립트에서 *Left* 함수를 사용하여 나온 출력을 보여 주는 Qlik Sense 테이블입니다.

텍스트	시작	Left
abcdef	3	abc
2021-07-14	4	2021
2021-07-14	2	20

*Index (page 1401)*를 참조하면 보다 복잡한 문자열 분석이 가능합니다.

Len

Len()은 입력 문자열의 길이를 반환합니다.

구문:

```
Len(text)
```

반환 데이터 유형: 정수

예: 차트 표현식

예	결과
`Len('Peter')`	'5'를 반환합니다.

예: 로드 스크립트

```
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 ];
```

결과

문자열	NewString
this is a sample text string	This is a sample text string
capitalize first letter only	Capitalize first letter only

LevenshteinDist

LevenshteinDist()는 두 문자열 사이의 Levenshtein 거리를 반환합니다. 이는 한 문자열을 다른 문자열로 변경하는 데 필요한 최소 단일 문자 편집 수(삽입, 삭제 또는 대체)로 정의됩니다. 이 함수는 퍼지 문자열 비교에 유용합니다.

구문:

```
LevenshteinDist(text1, text2)
```

반환 데이터 유형: 정수

예: 차트 표현식

예	결과
LevenshteinDist('Kitten','Sitting')	3을 반환합니다.

예: 로드 스크립트

로드 스크립트

```
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;
```

결과

ID	String_1	String_2	LevenshteinDistance
1	Silver	Sliver	2

ID	String_1	String_2	LevenshteinDistance
2	Silver	SSiver	2
3	Silver	SSiveer	3
4	Gold	굵게	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()는 입력 문자열의 모든 문자를 소문자로 변환합니다.

구문:

```
Lower(text)
```

반환 데이터 유형: 문자열

예: 차트 표현식

예	결과
`Lower('abcD')`	'abcd'를 반환합니다.

예: 로드 스크립트

```
Load  String, Lower(String) Inline [String rHode iSland washingTon d.C. new york];
```

결과

문자열	Lower(String)
rHode iSland	rhode island
washingTon d.C.	washington d.c.
new york	new york

LTrim

LTrim()은 모든 선행 공백이 제거된 입력 문자열을 반환합니다.

구문:

```
LTrim(text)
```

반환 데이터 유형: 문자열

예: 차트 표현식

예	결과
LTrim(' abc')	'abc'를 반환합니다.
LTrim('abc ')	'abc '를 반환합니다.

예: 로드 스크립트

```
Set verbatim=1;  T1:  Load *, len(LtrimString) as LtrimStringLength;       Load *, ltrim
(String) as LtrimString; Load *, len(String) as StringLength;                Load * Inline [
String '   abc    ' ' def  '];
```

> *"Set verbatim=1" 문이 예에 포함되어 있어 ltrim 함수의 데모를 보여 주기 전에 공백이 자동으로 잘리지 않습니다. 자세한 내용은 Verbatim (page 202)을 참조하십시오.*

결과

문자열	StringLength	LtrimStringLength
def	6	5
abc	10	7

관련 항목:

- *RTrim (page 1414)*

Mid

Mid()는 두 번째 인수 'start'로 정의된 문자의 위치에서 시작되고 세 번째 인수 'count'로 정의된 문자 수를 반환하는 입력 문자열의 일부를 반환합니다. 'count'를 생략하면 입력 문자열의 나머지가 반환됩니다. 입력 문자열의 첫 번째 문자는 번호가 1로 지정됩니다.

구문:

```
Mid(text, start[, count])
```

반환 데이터 유형: 문자열

인수:

인수

인수	설명
text	원래 문자열입니다.
start	text에서 포함할 첫 번째 문자의 위치를 정의하는 정수입니다.
count	출력 문자열의 길이를 정의합니다. 생략된 경우 **start**로 정의된 위치의 모든 문자가 포함됩니다.

예: 차트 표현식

예	결과
`Mid('abcdef',3 )`	'cdef'를 반환합니다.
`Mid('abcdef',3, 2 )`	'cd'를 반환합니다.

예: 로드 스크립트

```
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 ];
```

결과

로드 스크립트에서 *Mid* 함수를 사용하여 나온 출력을 보여 주는 Qlik Sense 테이블입니다.

텍스트	시작	Mid1	개수	Mid2
abcdef	2	bcdef	3	bcd
abcdef	3	cdef	2	cd
210714	2	10714	3	107
210714	3	0714	2	07

관련 항목:

Index (page 1401)

Ord

Ord()는 입력 문자열의 첫 번째 문자에 대한 Unicode 코드 포인트 번호를 반환합니다.

구문:

```
Ord(text)
```

반환 데이터 유형: 정수

예 및 결과:

차트 표현식

예	결과
Ord('A')	정수 65를 반환합니다.
Ord('Ab')	정수 65를 반환합니다.

로드 스크립트

```
//Guqin (Chinese: 古琴) – 7-stringed zithers T2: Load *,  ord(Chinese) as OrdUnicode,
        ord(Western) as OrdASCII;                       Load * inline [ Chinese, Western 古琴,
Guqin ];
```

결과:

Chinese	Western	OrdASCII	OrdUnicode
古琴	Guqin	71	21476

PurgeChar

PurgeChar()은 두 번째 인수('remove_chars')에 나타나는 문자를 제외하고, 입력 문자열('text')에 포함된 문자로 구성된 문자열을 반환합니다.

구문:

```
PurgeChar(text, remove_chars)
```

반환 데이터 유형: 문자열

인수:

인수

인수	설명
text	원래 문자열입니다.
remove_chars	제거할 text의 문자가 포함된 문자열입니다.

반환 데이터 유형: 문자열

예: 차트 표현식

예	결과
PurgeChar ('a1b2c3','123')	'abc'를 반환합니다.
PurgeChar ('a1b2c3','312')	'abc'를 반환합니다.

예: 로드 스크립트

```
T1:
Load
*,
purgechar(String1, String2) as PurgeChar;
Load * inline [
String1, String2
'a1b2c3', '123'
];
```

결과

로드 스크립트에서 *PurgeChar* 함수를 사용하여 나온 출력을 보여 주는 Qlik Sense 테이블

String1	String2	PurgeChar
a1b2c3	123	abc

관련 항목:

KeepChar (page 1404)

Repeat

Repeat()는 두 번째 인수로 정의된 횟수 만큼 반복된 입력 문자열로 구성된 문자열을 만듭니다.

구문:

```
Repeat(text[, repeat_count])
```

반환 데이터 유형: 문자열

인수:

인수

인수	설명
text	원래 문자열입니다.
repeat_count	문자열 **text**의 문자가 출력 문자열에서 반복되는 횟수를 정의합니다.

예: 차트 표현식

예	결과
`Repeat( ' * ', rating )` when **rating** = 4	'****'를 반환합니다.

예: 로드 스크립트

```
T1: Load *, repeat(String,2) as Repeat; Load * inline [ String hello world! hOw aRe you? ];
```

결과

문자열	반복
hello world!	hello world!hello world!
hOw aRe you?	hOw aRe you?hOw aRe you?

Replace

Replace()는 입력 문자열 내에서 특정 부분 문자열의 모든 발견 항목을 다른 부분 문자열로 대체한 후의 문자열을 반환합니다. 이 함수는 비재귀적이며 왼쪽에서 오른쪽으로 작동합니다.

구문:

```
Replace(text, from_str, to_str)
```

반환 데이터 유형: 문자열

인수:

인수

인수	설명
text	원래 문자열입니다.
from_str	입력 문자열 **text** 내에서 한 번 이상 발견될 수 있는 문자열입니다.
to_str	문자열 **text** 내 **from_str**의 모든 발견 항목을 대체할 문자열입니다.

예 및 결과:

예	결과
`Replace('abccde','cc','xyz')`	'abxyzde'를 반환합니다.

관련 항목:

Right

Right()는 입력 문자열의 마지막(가장 오른쪽) 문자로 구성된 문자열을 반환합니다. 문자 수는 두 번째 인수로 결정됩니다.

구문:

```
Right(text, count)
```

반환 데이터 유형: 문자열

인수:

인수

인수	설명
text	원래 문자열입니다.
count	문자열 **text**의 맨 오른쪽 부분부터 포함시킬 문자 수를 정의합니다.

예: 차트 표현식

예	결과
`Right('abcdef', 3)`	'def'를 반환합니다.

예: 로드 스크립트

```
T1:
Load
*,
right(Text,Start) as Right;
Load * inline [
Text, Start
'abcdef', 3
'2021-07-14', 4
'2021-07-14', 2
];
```

결과

로드 스크립트에서 *Right* 함수를 사용하여 나온 출력을 보여 주는 Qlik Sense 테이블

텍스트	시작	Right
abcdef	3	def
2021-07-14	4	7-14
2021-07-14	2	14

RTrim

RTrim()은 모든 후행 공백이 제거된 입력 문자열을 반환합니다.

구문:

```
RTrim(text)
```

반환 데이터 유형: 문자열

예: 차트 표현식

예	결과
`RTrim( ' abc' )`	' abc'를 반환합니다.
`RTrim( 'abc ' )`	'abc'를 반환합니다.

예: 로드 스크립트

```
Set verbatim=1;    T1:  Load *, len(RtrimString) as RtrimStringLength;         Load *, rtrim
(String) as RtrimString; Load *, len(String) as StringLength;              Load * Inline [
String '   abc    ' ' def  '];
```

"Set verbatim=1" 문이 예에 포함되어 있어 rtrim 함수의 데모를 보여 주기 전에 공백이 자동으로 잘리지 않습니다. 자세한 내용은 Verbatim (page 202)을 참조하십시오.

결과

문자열	StringLength	RtrimStringLength
def	6	4
abc	10	6

관련 항목:

LTrim (page 1409)

SubField

SubField()는 부모 문자열 필드에서 부분 문자열 성분을 추출하는 데 사용됩니다. 여기서 원래 레코드 필드는 구분 기호로 분리된 둘 이상의 부분으로 구성됩니다.

Subfield() 함수는 전체 이름으로 구성된 레코드 목록에서 이름과 성을 추출하거나 경로 이름의 구성 요소 부분 또는 쉼표로 구분된 테이블에서 데이터를 추출하는 등의 작업에 사용할 수 있습니다.

LOAD 문에서 옵션 field_no 파라메타를 생략하고 **Subfield()** 함수를 사용하면 각 부분 문자열에 대해 전체 레코드 하나가 생성됩니다. **Subfield()**를 사용하여 여러 필드를 로드하는 경우는 모든 조합의 카티션 곱이 생성됩니다.

구문:

```
SubField(text, delimiter[, field_no ])
```

반환 데이터 유형: 문자열

인수:

인수

인수	설명
text	원래 문자열입니다. 이 값은 하드 코딩된 텍스트, 변수, 달러 기호 확장 또는 다른 표현식이 될 수 있습니다.
delimiter	문자열을 구성 성분으로 나누는 입력 **text** 내의 문자열입니다.
field_no	세 번째 옵션 인수는 부모 문자열인 **text**의 부분 문자열이 반환되도록 지정하는 정수입니다. 첫 번째 부분 문자열을 반환하려면 값 1, 두 번째 부분 문자열을 반환하려면 2를 사용합니다. • **field_no**가 양수 값이면 부분 문자열이 왼쪽에서 오른쪽으로 추출됩니다. • **field_no**가 음수 값이면 부분 문자열이 오른쪽에서 왼쪽으로 추출됩니다.

SubField()는 Len(), Right(), Left(), Mid() 및 기타 문자열 함수를 복잡하게 조합한 함수를 대신하여 사용할 수 있습니다.

예: SubField를 사용한 스크립트 및 차트 표현식

예 - 스크립트 및 차트 표현식

기본 예

예	결과
`SubField(S, ';' ,2)`	**S**가 'abc;cde;efg'인 경우 'cde'를 반환합니다.
`SubField(S, ';' ,1)`	**S**가 빈 문자열인 경우 빈 문자열을 반환합니다.
`SubField(S, ';' ,1)`	**S**가 ';'인 경우 빈 문자열을 반환합니다.
경로 이름 vMyPath가 포함된 변수가 있다고 가정해 보겠습니다. `Set vMyPath=\Users\ext_jrb\Documents\Qlik\Sense\Apps;.`	텍스트 및 이미지 차트에서 다음과 같은 측정값을 추가할 수 있습니다. `SubField(vMyPath, '\',-3)`, 결과 값은 'Qlik'으로, 이 값이 변수 vMyPath 오른쪽 끝에서 세 번째의 하위 문자열이기 때문입니다.

스크립트 예 1

로드 스크립트

데이터 로드 편집기에서 다음 스크립트 표현식과 데이터를 로드합니다.

```
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;
```

시각화 만들기

Name, **FirstName** 및 **SurName**을 차원으로 사용하여 Qlik Sense 시트에 테이블 시각화를 만듭니다.

결과

Name	FirstName	SurName
Dave Owen	Dave	Owen
Joe Tem	Joe	Tem

설명

SubField() 함수는 **field_no** 인수를 1로 설정하여 **Name**의 첫 번째 부분 문자열을 추출합니다. **field_no**의 값이 양수이므로 하위 열을 추출하기 위해 왼쪽에서 오른쪽 순서를 따릅니다. 두 번째 함수 호출은 **field_no** 인수를 -1로 설정하여 두 번째 하위 문자열을 추출합니다. 그러면 오른쪽에서 왼쪽 순서로 하위 문자열이 추출됩니다.

스크립트 예 2

로드 스크립트

데이터 로드 편집기에서 다음 스크립트 표현식과 데이터를 로드합니다.

```
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 '|');
```

시각화 만들기

Instrument, **Player** 및 **Project**를 차원으로 사용하여 Qlik Sense 시트에 테이블 시각화를 만듭니다.

결과

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
Synth	Neil	Music
Synth	Neil	Video
Synth	Neil	OST

설명

이 예는 **Subfield()** 함수의 여러 인스턴스를 사용하는 방법을 보여줍니다. 각각은 field_no 파라메타가 생략되어 있으며 동일한 **LOAD** 문 내에서 모든 조합의 카티션 곱을 생성합니다. **DISTINCT** 옵션은 중복 레코드 생성을 피하기 위해 사용됩니다.

SubStringCount

SubStringCount()는 입력 문자열 텍스트에 지정된 부분 문자열의 발생 횟수를 반환합니다. 일치하는 항목이 없으면 0이 반환됩니다.

구문:

```
SubStringCount(text, sub_string)
```

반환 데이터 유형: 정수

인수:

인수	설명
text	원래 문자열입니다.
sub_string	입력 문자열 **text** 내에서 한 번 이상 발견될 수 있는 문자열입니다.

예: 차트 표현식

예	결과
`SubStringCount ( 'abcdefgcdxyz', 'cd' )`	'2'를 반환합니다.
`SubStringCount ( 'abcdefgcdxyz', 'dc' )`	'0'을 반환합니다.

예: 로드 스크립트

```
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 ];
```

결과

문자열	SubStringCount_AB
aB/cd/ef/gh/Abc/abandoned	3
ABC:DEF:GHI:AB:CD:EF:GH	2

TextBetween

TextBetween()은 구분 기호로 지정된 문자 사이에서 발견되는 입력 문자열의 텍스트를 반환합니다.

구문:

```
TextBetween(text, delimiter1, delimiter2[, n])
```

반환 데이터 유형: 문자열

인수:

인수	설명
text	원래 문자열입니다.
delimiter1	**text**에서 검색할 첫 번째 구분 문자(또는 문자열)를 지정합니다.
delimiter2	**text**에서 검색할 두 번째 구분 문자(또는 문자열)를 지정합니다.
n	검색 범위의 기준이 되는 구분 기호 쌍의 발생 위치를 정의합니다. 예를 들어, 값 2를 지정하면 delimiter1의 두 번째 발생 항목과 delimiter2의 두 번째 발생 항목 사이의 문자를 반환합니다.

예: 차트 표현식

예	결과
`TextBetween('<abc>', '<', '>')`	'abc'를 반환합니다.
`TextBetween('<abc><de>', '<', '>',2)`	'de'를 반환합니다.

예	결과
`TextBetween('abc', '<', '>')` `TextBetween('<a<b', '<', '>')`	두 예 모두 NULL을 반환합니다. 문자열에 구분 기호가 없는 경우 NULL이 반환됩니다.
`TextBetween('<>', '<', '>')`	빈 문자열을 반환합니다.
`TextBetween('<abc>', '<', '>', 2)`	n이 구분 기호의 발생 횟수보다 크므로 NULL을 반환합니다.

예: 로드 스크립트

```
Load *, textbetween(Text,'<','>') as TextBetween, textbetween(Text,'<','>',2) as
SecondTextBetween; Load * inline [ Text <abc><de> <def><ghi><jkl> ];
```

결과

텍스트	TextBetween	SecondTextBetween
<abc><de>	abc	de
<def><ghi><jkl>	def	ghi

Trim

Trim()은 모든 선행 및 후행 공백이 제거된 입력 문자열을 반환합니다.

구문:

```
Trim(text)
```

반환 데이터 유형: 문자열

예 및 결과:

차트 표현식

예	결과
`Trim( ' abc' )`	'abc'를 반환합니다.
`Trim( 'abc ' )`	'abc'를 반환합니다.
`Trim( ' abc ' )`	'abc'를 반환합니다.

로드 스크립트

```
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');
```

"Set verbatim=1" 문이 예에 포함되어 있어 trim 함수의 데모를 보여 주기 전에 공백이 자동으로 잘리지 않습니다. 자세한 내용은 Verbatim (page 202)을 참조하십시오.

결과:

문자열	StringLength	TrimStringLength
def	6	3
abc	10	3

Upper

Upper()는 표현식의 모든 텍스트 문자에 대해 입력 문자열의 모든 문자를 대문자로 변환합니다. 숫자와 기호는 무시됩니다.

구문:

```
Upper(text)
```

반환 데이터 유형: 문자열

예: 차트 표현식

예	결과
`Upper(' abcD')`	'ABCD'를 반환합니다.

예: 로드 스크립트

```
Load  String,Upper(String) Inline [String rHode iSland washingTon d.C. new york];
```

결과

문자열	Upper(String)
rHode iSland	RHODE ISLAND
washingTon d.C.	WASHINGTON D.C.
new york	NEW YORK

8.25 시스템 함수

시스템 함수는 시스템, 장치 및 Qlik Sense 앱 속성에 액세스하기 위한 함수를 제공합니다.

시스템 함수 개요

일부 함수는 개요가 끝난 후에 더 자세히 설명합니다. 이들 함수는 구문에서 함수 이름을 클릭하여 해당 함수에 대한 상세 설명에 즉시 액세스할 수도 있습니다.

Author()
이 함수는 현재 앱의 작성자 속성이 포함된 문자열을 반환합니다. 데이터 로드 스크립트와 차트 표현식 모두에서 사용할 수 있습니다.

Qlik Sense의 현재 버전에서는 작성자 속성을 설정할 수 없습니다. QlikView 문서를 마이그레이션 하면 작성자 속성이 유지됩니다.

ClientPlatform()
이 함수는 클라이언트 브라우저의 사용자 에이전트 문자열을 반환합니다. 데이터 로드 스크립트와 차트 표현식 모두에서 사용할 수 있습니다.

```
Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/35.0.1916.114 Safari/537.36
```

ComputerName
이 함수는 운영 체제에서 반환한 대로 컴퓨터의 이름이 포함된 문자열을 반환합니다. 데이터 로드 스크립트와 차트 표현식 모두에서 사용할 수 있습니다.

컴퓨터 이름이 15자보다 큰 경우 문자열에는 처음 15자만 포함됩니다.

```
ComputerName( )
```

DocumentName
이 함수는 경로는 제외하되 확장명을 포함하여 현재 Qlik Sense 앱의 이름이 포함된 문자열을 반환합니다. 데이터 로드 스크립트와 차트 표현식 모두에서 사용할 수 있습니다.

```
DocumentName( )
```

DocumentPath
이 함수는 현재 Qlik Sense 앱의 전체 경로가 포함된 문자열을 반환합니다. 데이터 로드 스크립트와 차트 표현식 모두에서 사용할 수 있습니다.

```
DocumentPath( )
```

표준 모드에서는 이 기능이 지원되지 않습니다.

DocumentTitle
이 함수는 현재 Qlik Sense 앱의 제목이 포함된 문자열을 반환합니다. 데이터 로드 스크립트와 차트 표현식 모두에서 사용할 수 있습니다.

```
DocumentTitle( )
```

EngineVersion
이 함수는 전체 Qlik Sense 엔진 버전을 문자열로 반환합니다.

```
EngineVersion ()
```

GetCollationLocale
이 스크립트 함수는 사용된 국가별 로캘의 문화권 이름을 반환합니다. CollationLocale 변수를 설정하지 않은 경우 실제 사용자 시스템의 로캘이 반환됩니다.

```
GetCollationLocale( )
```

GetObjectField
GetObjectField()는 차원의 이름을 반환합니다. **Index**는 어떤 차원을 반환해야 하는지 지정하는 선택적 정수입니다.

```
GetObjectField - 차트 함수([index])
```

GetRegistryString
이 함수는 Windows 레지스트리의 키 값을 반환합니다. 데이터 로드 스크립트와 차트 표현식 모두에서 사용할 수 있습니다.

```
GetRegistryString(path, key)
```

표준 모드에서는 이 기능이 지원되지 않습니다.

GetSysAttr
이 함수는 선택한 앱의 테넌트 및 공간 도메인 특성을 반환합니다. 데이터 로드 스크립트와 차트 표현식 모두에서 사용할 수 있습니다.

```
GetSysAttr (name)
```

Qlik Sense 클라이언트 관리에서 이 함수를 사용하면 빈 데이터 값만 반환됩니다.

IsPartialReload
이 함수는 현재 다시 로드가 부분 다시 로드일 경우 -1(True)을, 그렇지 않으면 0(False)을 반환합니다.

```
IsPartialReload ()
```

InObject
InObject() 차트 함수는 현재 개체가 함수 인수에 지정된 ID를 가진 다른 개체 안에 포함되어 있는지 여부를 평가합니다. 개체는 시트 또는 시각화일 수 있습니다.

```
InObject - 차트 함수(id_str)
```

ObjectId
ObjectId() 차트 함수는 표현식이 평가되는 개체의 ID를 반환합니다.이 함수는 함수와 관련된 개체 유형을 지정하는 선택적 인수를 사용합니다. 개체는 시트 또는 시각화일 수 있습니다. 이 함수는 차트 표현식에서만 사용할 수 있습니다.

```
ObjectId - 차트 함수([object_type_str])
```

OSUser
이 함수는 현재 연결된 사용자의 이름이 포함된 문자열을 반환합니다. 데이터 로드 스크립트와 차트 표현식 모두에서 사용할 수 있습니다.

```
OSUser( )
```

Qlik Sense Desktop 및 Qlik Sense Client-Managed 모바일에서 이 함수는 항상 'Personal\Me'를 반환합니다.

ProductVersion
이 함수는 전체 Qlik Sense 버전 및 빌드 번호를 문자열로 반환합니다.

이 함수는 사용이 중단되고 **EngineVersion()**으로 대체됩니다.

```
ProductVersion ()
```

ReloadTime
이 함수는 마지막으로 데이터 로드를 완료했을 때의 타임스탬프를 반환합니다. 데이터 로드 스크립트와 차트 표현식 모두에서 사용할 수 있습니다.

```
ReloadTime( )
```

StateName
StateName()은 해당 함수가 사용되는 시각화의 대체 상태 이름을 반환합니다. 예를 들어 StateName을 사용하여 동적 텍스트 및 색으로 시각화를 만들어 시각화 상태가 변경되었음을 반영할 수 있습니다. 차트 식에서 이 함수를 사용할 수 있지만 식이 나타내는 상태를 결정하는 데에는 사용할 수 없습니다.

```
StateName - 차트 함수()
```

EngineVersion

이 함수는 전체 Qlik Sense 엔진 버전을 문자열로 반환합니다.

구문:

```
EngineVersion()
```

GetSysAttr

이 함수는 선택한 앱의 테넌트 및 공간 도메인 특성을 반환합니다. 데이터 로드 스크립트와 차트 표현식 모두에서 사용할 수 있습니다.

Qlik Sense 클라이언트 관리에서 이 함수를 사용하면 빈 데이터 값이 반환됩니다. 따라서 나중에 앱을 Qlik Cloud에 업로드하기 위해 이 함수를 사용하여 오류 발생 없이 Qlik Sense 클라이언트 관리에서 로드 스크립트를 개발할 수 있습니다.

Qlik Cloud 함수에 대한 전체 문서에 액세스하려면 GetSysAttr - 스크립트 및 차트 함수를 참조하십시오.

InObject - 차트 함수

InObject() 차트 함수는 현재 개체가 함수 인수에 지정된 ID를 가진 다른 개체 안에 포함되어 있는지 여부를 평가합니다. 개체는 시트 또는 시각화일 수 있습니다.

이 함수는 최상위 시트 개체에서 다른 시각화 내에 중첩된 시각화에 이르기까지 시트의 개체 계층 구조를 표시하는 데 사용할 수 있습니다. 이 함수는 **if** 및 **ObjectId** 함수와 함께 사용하여 앱에서 사용자 지정 탐색을 만들 수 있습니다.

구문:

```
InObject(id_str)
```

반환 데이터 유형: 부울

Qlik Sense에서 부울 true 값은 -1로 표시되고 false 값은 0으로 표시됩니다.

인수

인수	설명
id_str	평가 중인 개체의 ID를 나타내는 문자열 값입니다.

시트 ID는 앱 URL에서 가져올 수 있습니다. 시각화의 경우 **개발자** 옵션을 사용하여 개체 ID와 개체 유형의 텍스트 문자열을 식별합니다.

다음과 같이 하십시오.

1. 분석 모드에서 URL에 다음 텍스트를 추가합니다.
   /options/developer
2. 시각화를 마우스 오른쪽 버튼으로 클릭하고 **개발자**를 클릭합니다.
3. **속성**에서 대화상자 헤더의 개체 ID와 **"qType"** 속성의 개체 유형을 가져옵니다.

제한 사항:

이 함수는 마스터 항목인 컨테이너 내부의 개체(예: 버튼)에서 호출될 때 예기치 않은 결과를 제공할 수 있습니다. 이 제한은 여러 목록 상자의 컨테이너인 필터 창 마스터 항목에도 적용됩니다. 이는 마스터 항목이 개체 계층 구조를 사용하는 방식 때문입니다.

InObject()는 종종 다음 함수와 함께 사용됩니다.

관련 함수

함수	상호 작용
if (page 540)	**if** 및 **ObjectId** 함수를 함께 사용하여 조건부 표현식을 만들 수 있습니다. 예를 들어, 시각화는 이러한 함수를 사용하는 표현식을 통해 조건부 색을 얻을 수 있습니다.
ObjectId - 차트 함수 (page 1429)	**if**와 마찬가지로 **ObjectId**도 **InObject**와 함께 사용되어 조건부 표현식을 만듭니다.

예 1 – 기본 기능

차트 표현식 및 결과

다음 기본 예에서는 개체가 다른 개체 안에 포함되어 있는지 여부를 확인하는 방법을 보여 줍니다. 이 경우 **텍스트 및 이미지** 개체는 시트의 ID를 인수로 사용하는 시트 개체에 상주합니다.

다음과 같이 하십시오.

1. 새 시트를 열고 **텍스트 및 이미지** 차트를 시트에 넣습니다.
2. 속성 패널에서 **측정값 추가**를 클릭합니다.
3. *fx*를 클릭하여 수식 편집기를 엽니다.
4. 다음 표현식을 대화 상자에 붙여넣습니다.
   ```
   =InObject()
   ```
5. 시트의 ID를 괄호 사이에 문자열로 포함하도록 표현식을 수정합니다.
   예를 들어, ID가 1234-5678인 시트의 경우 다음을 사용합니다.
   ```
   =InObject('1234-5678')
   ```
6. **적용**을 클릭합니다.

-1 값이 차트에 표시되어 표현식이 true로 평가되었음을 나타냅니다.

예 2 – 조건부 색이 있는 개체

차트 표현식 및 결과

개요

다음 예에서는 현재 열려 있는 시트를 나타내기 위해 다양한 색을 표시하는 사용자 지정 탐색 버튼을 만드는 방법을 보여 줍니다.

새 앱을 만들고 데이터 로드 편집기를 열어 시작합니다. 다음 로드 스크립트를 새 탭에 붙여넣습니다. 데이터 자체는 자리 표시자이며 콘텐츠 예에서는 사용되지 않습니다.

로드 스크립트

```
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
];
```

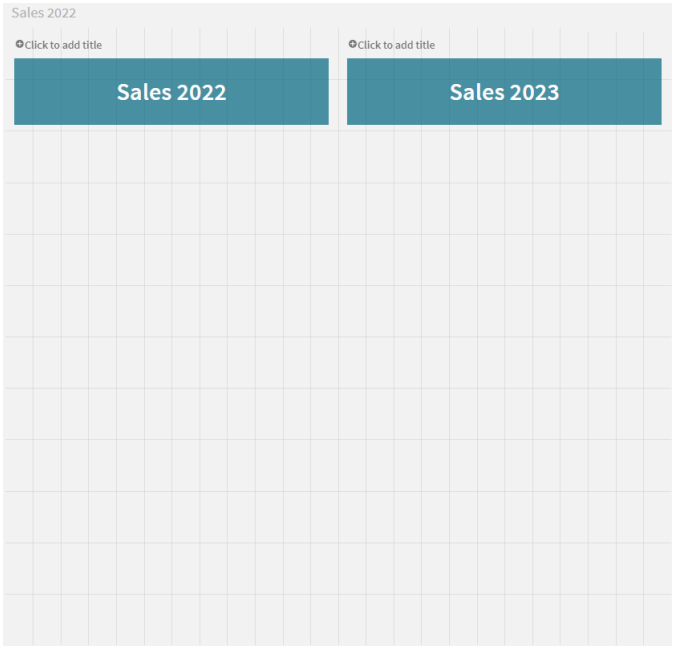
시각화 만들기

데이터를 로드하고 두 개의 새 시트를 만듭니다. 각각 *Sales 2022* 및 *Sales 2023*이라는 제목을 지정합니다.

다음으로 두 시트 사이를 탐색하는 데 사용할 두 개의 버튼 개체를 만듭니다.

다음과 같이 하십시오.

1. 두 개의 **버튼** 개체를 시트에 추가합니다.
2. **모양** > **일반**, 각 버튼의 **레이블**을 각각 *Sales 2022* 및 *Sales 2023*으로 설정합니다.
3. 다음 이미지와 일치하도록 버튼을 정렬합니다.
   두 개의 탐색 버튼이 있는 Sales 2022 시트 배열



4. *Sales 2022* 버튼을 선택하고 속성 패널에서 **작업 및 탐색**을 펼칩니다.
5. **작업 추가**를 클릭하고 **탐색**에서 **시트로 이동**을 선택합니다.
6. **시트**에서 *Sales 2022*를 선택합니다.
7. 이 버튼 작업 설정을 반복하여 **Sales 2023** 버튼을 *Sales 2023* 시트에 연결합니다.
8. 버튼을 마우스 오른쪽 버튼으로 클릭하고 **마스터 항목에 추가**를 선택하여 버튼을 마스터 항목으로 변환합니다.

이제 시트에서 동일한 크기와 배열을 사용하여 각 버튼을 복사하여 *Sales 2023* 시트에 붙여넣을 수 있습니다.

조건부 색 만들기

다음으로 버튼이 현재 열려 있는 시트에 연결된 경우 파란색이 되고 열려 있지 않은 시트에 연결된 경우 밝은 회색이 되도록 버튼을 구성합니다.

다음과 같이 하십시오.

1. *Sales 2022* 시트를 열고 URL에서 시트 ID를 가져옵니다. *Sales 2022* 시트를 열어 둡니다.
2. **Sales 2022** 버튼 마스터 항목을 클릭하고 속성 패널에서 **편집**을 선택합니다.
3. **모양** > **배경**에서 선택하여 **표현식별** 버튼 색을 지정합니다.
4. **표현식**에 다음 텍스트를 붙여넣습니다.

=if(InObject(''), Blue(), LightGray())

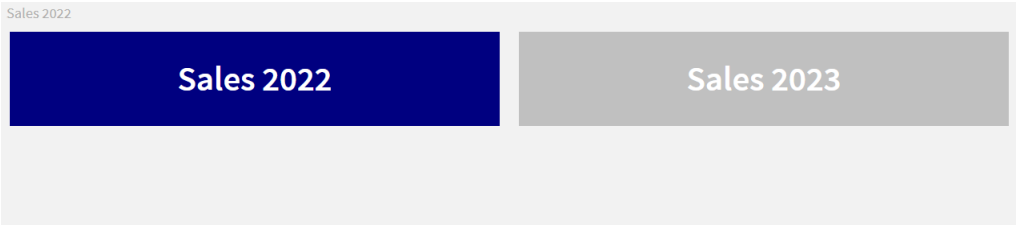
5. 위 표현식의 괄호 사이에 *Sales 2022* 시트의 시트 ID를 붙여넣습니다.

이제 *Sales 2022* 시트가 열려 있으면 버튼이 파란색으로 바뀌고 열려 있지 않으면 밝은 회색으로 바뀌도록 구성되어 있습니다.

Sales 2023 시트에 대한 위 지침을 반복하여 **Sales 2023** 버튼 마스터 항목을 *Sales 2023* 시트 ID에 연결합니다.

이제 각 시트에는 파란색으로 현재 열려 있는 시트를 나타내는 두 개의 버튼이 있습니다.

현재 Sales 2022가 표시되어 있음을 나타내도록 파란색으로 지정된 Sales 2022 시트



IsPartialReload

이 함수는 현재 다시 로드가 부분 다시 로드일 경우 -1(True)을, 그렇지 않으면 0(False)을 반환합니다.

구문:

```
IsPartialReload()
```

ObjectId - 차트 함수

ObjectId() 차트 함수는 표현식이 평가되는 개체의 ID를 반환합니다.이 함수는 함수와 관련된 개체 유형을 지정하는 선택적 인수를 사용합니다. 개체는 시트 또는 시각화일 수 있습니다. 이 함수는 차트 표현식에서만 사용할 수 있습니다.

구문:

```
ObjectId([object_type_str])
```

반환 데이터 유형: 문자열

함수의 유일한 인수인 **object_type_str**은 선택 사항이며 개체의 유형을 나타내는 문자열 값을 나타냅니다.

인수

인수	설명
object_type_str	평가 중인 개체의 유형을 나타내는 문자열 값입니다.

함수 표현식에 인수가 지정되지 않은 경우 **ObjectId()**는 표현식이 사용된 개체의 ID를 반환합니다. 시각화가 표시되는 시트 개체의 ID를 반환하려면 *ObjectId('sheet')*를 사용합니다.

다른 시각화 개체 내에 중첩된 시각화 개체의 경우 다른 결과를 위해 함수 인수에 원하는 개체 유형을 지정합니다. 예를 들어, 컨테이너 내 **텍스트 및 이미지** 차트에서 *'text-image'*를 사용하여 **텍스트 및 이미지** 개체 및 *'container'*는 컨테이너의 ID를 반환합니다.

다음과 같이 하십시오.

1. 분석 모드에서 URL에 다음 텍스트를 추가합니다.
   /options/developer
2. 시각화를 마우스 오른쪽 버튼으로 클릭하고 **개발자**를 클릭합니다.
3. **속성**에서 대화상자 헤더의 개체 ID와 **"qType"** 속성의 개체 유형을 가져옵니다.

제한 사항:

이 함수는 마스터 항목인 컨테이너 내부의 개체(예: 버튼)에서 호출될 때 예기치 않은 결과를 제공할 수 있습니다. 이 제한은 여러 목록 상자의 컨테이너인 필터 창 마스터 항목에도 적용됩니다. 이는 마스터 항목이 개체 계층 구조를 사용하는 방식 때문입니다.

이러한 경우 차트 표현식 *ObjectId('sheet')*는 빈 문자열을 반환하지만 *ObjectId('masterobject')*는 소유한 마스터 항목의 식별자를 표시합니다.

ObjectId()는 종종 다음 함수와 함께 사용됩니다.

관련 함수

함수	상호 작용
if (page 540)	**if** 및 **ObjectId** 함수를 함께 사용하여 조건부 표현식을 만들 수 있습니다. 예를 들어, 시각화는 이러한 함수를 사용하는 표현식을 통해 조건부 색을 얻을 수 있습니다.
InObject - 차트 함수 (page 1425)	**if**와 마찬가지로 **InObject**도 **ObjectId**와 함께 사용되어 조건부 표현식을 만듭니다.

예 1 - 차트 개체 ID 반환

차트 표현식 및 결과
다음 기본 예는 시각화의 ID를 반환하는 방법을 보여 줍니다.

다음과 같이 하십시오.

1. 새 시트를 열고 **텍스트 및 이미지** 차트를 시트에 넣습니다.
2. 속성 패널에서 **측정값 추가**를 클릭합니다.
3. fx를 클릭하여 수식 편집기를 엽니다.
4. 다음 표현식을 대화 상자에 붙여넣습니다.
   ```
   =ObjectId()
   ```
5. **적용**을 클릭합니다.

텍스트 및 이미지 개체가 시각화에 표시됩니다.

다음 표현식으로도 동일한 결과를 얻을 수 있습니다.

```
=ObjectId('text-image')
```

예 2 – 시트 ID 반환

차트 표현식 및 결과
다음 기본 예는 시각화가 표시되는 시트의 ID를 반환하는 방법을 보여 줍니다.

다음과 같이 하십시오.

1. 새 시트를 열고 **텍스트 및 이미지** 차트를 시트에 넣습니다.
2. 속성 패널에서 **측정값 추가**를 클릭합니다.
3. fx를 클릭하여 수식 편집기를 엽니다.
4. 다음 표현식을 대화 상자에 붙여넣습니다.
   ```
   =ObjectId('sheet')
   ```
5. **적용**을 클릭합니다.

시트의 ID가 시각화에 표시됩니다.

예 3 - 중첩 표현식

차트 표현식 및 결과
다음 예에서는 **ObjectId()** 함수가 다른 표현식 내에 중첩될 수 있는 방법을 보여 줍니다.

다음과 같이 하십시오.

1. 새 시트를 열고 **텍스트 및 이미지** 차트를 시트에 넣습니다.
2. 속성 패널에서 **측정값 추가**를 클릭합니다.
3. fx를 클릭하여 수식 편집기를 엽니다.

4. 다음 표현식을 대화 상자에 붙여넣습니다.
   =if(InObject(ObjectId('text-image')), 'In Text & image', 'Not in Text & image')
5. **적용**을 클릭합니다.

텍스트 *In Text & image*가 차트에 표시되어 표현식에서 참조하는 개체가 **Text & image** 차트입니다.

조건부 색을 사용하는 자세한 예는 *InObject - 차트 함수 (page 1425)*의 예를 참조하십시오.

ProductVersion

이 함수는 전체 Qlik Sense 버전 및 빌드 번호를 문자열로 반환합니다. 이 함수는 사용이 중단되고 **EngineVersion()**으로 대체됩니다.

구문:

```
ProductVersion()
```

StateName - 차트 함수

StateName()은 해당 함수가 사용되는 시각화의 대체 상태 이름을 반환합니다. 예를 들어 StateName을 사용하여 동적 텍스트 및 색으로 시각화를 만들어 시각화 상태가 변경되었음을 반영할 수 있습니다. 차트 식에서 이 함수를 사용할 수 있지만 식이 나타내는 상태를 결정하는 데에는 사용할 수 없습니다.

구문:

```
StateName ()
```

Example 1:

동적 텍스트

```
='Region - ' & if(StateName() = '$', 'Default', StateName())
```

Example 2:

동적 색

```
if(StateName() = 'Group 1', rgb(152, 171, 206),
      if(StateName() = 'Group 2', rgb(187, 200, 179),
            rgb(210, 210, 210)
      )
)
```

8.26 테이블 함수

테이블 함수는 현재 읽는 중인 데이터 테이블에 대한 정보를 반환합니다. 테이블 이름을 지정하지 않고 **LOAD** 문 내에서 이 함수를 사용하면 현재 테이블이 사용됩니다.

모든 함수를 데이터 로드 스크립트에서 사용할 수 있는 반면, **NoOfRows**는 차트 표현식에서만 사용할 수 있습니다.

테이블 함수 개요

일부 함수는 개요가 끝난 후에 더 자세히 설명합니다. 이들 함수는 구문에서 함수 이름을 클릭하여 해당 함수에 대한 상세 설명에 즉시 액세스할 수도 있습니다.

FieldName

FieldName 스크립트 함수는 이전에 로드된 테이블 내에서 지정된 숫자가 포함된 필드의 이름을 반환합니다. 이 함수를 **LOAD** 문 내에서 사용하는 경우 현재 로드되고 있는 테이블을 참조하지 않도록 해야 합니다.

```
FieldName (field_number ,table_name)
```

FieldNumber

FieldNumber 스크립트 함수는 이전에 로드된 테이블 내에서 지정된 필드의 수를 반환합니다. 이 함수를 **LOAD** 문 내에서 사용하는 경우 현재 로드되고 있는 테이블을 참조하지 않도록 해야 합니다.

```
FieldNumber (field_name ,table_name)
```

NoOfFields

NoOfFields 스크립트 함수는 이전에 로드된 테이블 내의 필드 수를 반환합니다. 이 함수를 **LOAD** 문 내에서 사용하는 경우 현재 로드되고 있는 테이블을 참조하지 않도록 해야 합니다.

```
NoOfFields (table_name)
```

NoOfRows

NoOfRows 함수는 이전에 로드된 테이블의 행(레코드) 수를 반환합니다. 이 함수를 **LOAD** 문 내에서 사용하는 경우 현재 로드되고 있는 테이블을 참조하지 않도록 해야 합니다.

```
NoOfRows (table_name)
```

NoOfTables

이 스크립트 함수는 이전에 로드된 테이블의 수를 반환합니다.

```
NoOfTables()
```

TableName

이 스크립트 함수는 지정된 숫자가 포함된 테이블의 이름을 반환합니다.

```
TableName(table_number)
```

TableNumber

이 스크립트 함수는 지정된 테이블의 수를 반환합니다. 첫 번째 테이블 번호는 0입니다.

table_name이 존재하지 않으면 NULL이 반환됩니다.

```
TableNumber(table_name)
```

이 예에서는 로드한 테이블 및 필드에 대한 정보가 포함된 테이블을 만들려고 합니다.

우선 샘플 데이터를 로드합니다. 그러면 이 섹션에 나온 테이블 함수를 설명하는 데 사용할 두 테이블이 생성됩니다.

```
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 ;
```

다음으로, 로드된 테이블을 **NoOfTables** 함수를 사용하여 반복하고 각 테이블의 필드를 **NoOfFields** 함수를 사용하여 반복하고 테이블 함수를 사용하여 정보를 로드합니다.

```
//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;
```

결과 테이블 Tables는 다음과 같습니다.

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

FieldName 스크립트 함수는 이전에 로드된 테이블 내에서 지정된 숫자가 포함된 필드의 이름을 반환합니다. 이 함수를 **LOAD** 문 내에서 사용하는 경우 현재 로드되고 있는 테이블을 참조하지 않도록 해야 합니다.

구문:

```
FieldName(field_number ,table_name)
```

인수:

인수

인수	설명
field_number	참조할 필드의 필수 번호입니다.
table_name	참조할 필드가 포함된 테이블입니다.

```
LET a = FieldName(4,'tab1');
```

FieldNumber

FieldNumber 스크립트 함수는 이전에 로드된 테이블 내에서 지정된 필드의 수를 반환합니다. 이 함수를 **LOAD** 문 내에서 사용하는 경우 현재 로드되고 있는 테이블을 참조하지 않도록 해야 합니다.

구문:

```
FieldNumber(field_name ,table_name)
```

인수:

인수

인수	설명
field_name	필드의 이름입니다.
table_name	필드가 포함된 테이블의 이름입니다.

field_name 필드가 table_name에 존재하지 않거나 table_name이 존재하지 않는 경우 이 함수는 0을 반환합니다.

```
LET a = FieldNumber('Customer','tab1');
```

NoOfFields

NoOfFields 스크립트 함수는 이전에 로드된 테이블 내의 필드 수를 반환합니다. 이 함수를 **LOAD** 문 내에서 사용하는 경우 현재 로드되고 있는 테이블을 참조하지 않도록 해야 합니다.

구문:

```
NoOfFields(table_name)
```

인수:

인수

인수	설명
table_name	테이블의 이름입니다.

```
LET a = NoOfFields('tab1');
```

NoOfRows

NoOfRows 함수는 이전에 로드된 테이블의 행(레코드) 수를 반환합니다. 이 함수를 **LOAD** 문 내에서 사용하는 경우 현재 로드되고 있는 테이블을 참조하지 않도록 해야 합니다.

구문:

```
NoOfRows(table_name)
```

인수:

인수

인수	설명
table_name	테이블의 이름입니다.

```
LET a = NoOfRows('tab1');
```

8.27 삼각 함수 및 쌍곡선 함수

이 섹션에서는 삼각법 및 쌍곡선 연산을 수행하는 함수에 대해 설명합니다. 모든 함수에서 인수는 라디안으로 측정된 각도로 평가되는 표현식이며, 여기서 **x**는 실수로 해석되어야 합니다.

모든 각도는 라디안으로 측정합니다.

모든 함수는 데이터 로드 스크립트와 차트 표현식 모두에서 사용할 수 있습니다.

cos
x의 코사인입니다. 결과는 -1 ~ 1 사이의 숫자입니다.

```
cos( x )
```

acos
x의 역코사인입니다. 이 함수는 $-1 \le x \le 1$인 경우에만 정의됩니다. 결과는 0 ~ π 사이의 숫자입니다.

```
acos( x )
```

sin
x의 사인입니다. 결과는 -1 ~ 1 사이의 숫자입니다.

```
sin( x )
```

asin
x의 역사인입니다. 이 함수는 -1≤**x**≤1인 경우에만 정의됩니다. 결과는 - π/2 ~ π/2 사이의 숫자입니다.

```
asin( x )
```

tan
x의 탄젠트입니다. 결과는 실수입니다.

```
tan( x )
```

atan
x의 역탄젠트입니다. 결과는 - π/2 ~ π/2 사이의 숫자입니다.

```
atan( x )
```

atan2
역탄젠트 함수의 2차원 일반화입니다. 원점과 **x** 및 **y** 좌표가 가리키는 지점 간의 각도를 반환합니다. 결과는 - π ~ + π 사이의 숫자입니다.

```
atan2( y,x )
```

cosh
x의 쌍곡선 코사인입니다. 결과는 양의 실수입니다.

```
cosh( x )
```

sinh
x의 쌍곡선 사인입니다. 결과는 실수입니다.

```
sinh( x )
```

tanh
x의 쌍곡선 탄젠트입니다. 결과는 실수입니다.

```
tanh( x )
```

acosh
x의 역쌍곡선 코사인입니다. 결과는 양의 실수입니다.

```
acosh( x )
```

asinh
x의 역쌍곡선 사인입니다. 결과는 실수입니다.

```
asinh( x )
```

atanh
x의 역쌍곡선 탄젠트입니다. 결과는 실수입니다.

```
atanh( x )
```

다음 스크립트 코드는 샘플 테이블을 로드한 후 값에 대해 계산된 삼각법 및 쌍곡선 연산이 포함된 테이블을 로드합니다.

```
SampleData:
LOAD * Inline
[Value
-1
0
1];

Results:
Load *,
cos(Value),
acos(Value),
sin(Value),
asin(Value),
tan(Value),
atan(Value),
atan2(Value, Value),
cosh(Value),
sinh(Value),
tanh(Value)
RESIDENT SampleData;

Drop Table SampleData;
```

8.28 창 함수

창 함수는 여러 행의 값을 사용하여 계산을 수행하여 각 행에 대한 값을 별도로 생성합니다. 창 함수는 전체 테이블을 읽은 후에만 계산할 수 있습니다.

창 함수를 사용하여 다음과 같은 작업을 수행할 수 있습니다.

- 행의 개별 숫자 값을 열 내의 평균, 최댓값 또는 최솟값과 비교합니다.
- 열 내에서 또는 전체 테이블 내에서 개별 값의 순위를 계산합니다.

창 함수는 테이블의 레코드 수를 변경하지 않지만 집계 함수 또는 관계 함수 및 범위 함수와 유사한 작업을 수행할 수 있습니다.

각 함수는 개요가 끝난 후에 더 자세히 설명합니다. 구문에서 함수 이름을 클릭하여 해당 함수에 대한 상세 설명에 즉시 액세스할 수도 있습니다.

Window

Window 함수는 여러 행에서 계산을 수행하여 각 행에 대한 값을 개별적으로 생성합니다.

```
Window - 스크립트 함수:(input_expr, [partition1, partition2, ...], [sort_type,
[sort_expr]],[filter_expr], [start_expr,end_expr])[row_window_size])
```

WRank
WRank 함수는 **Window** 내에서 순위 계산을 수행합니다.

```
WRank - 스크립트 함수:([TOTAL] expr[, mode[, fmt]])
```

Window - 스크립트 함수:

Window()는 여러 행에서 계산을 수행하여 각 행에 대한 값을 개별적으로 생성합니다.

Window 함수를 사용하여 다음과 같은 작업을 수행할 수 있습니다.

- 행의 개별 숫자 값을 열 내의 평균, 최댓값 또는 최솟값과 비교합니다.
- 열 내에서 또는 전체 테이블 내에서 개별 값의 순위를 계산합니다.

Window 함수는 테이블의 레코드 수를 변경하지 않지만 집계, 관계형 및 범위 함수와 유사한 작업을 계속 수행할 수 있습니다.

Window 함수는 테이블에 추가하려면 작업 중인 테이블의 LOAD 문 내에 캐시가 있어야 합니다. 예:

```
[Transactions]:
Load
       *,
       Window(avg(Expression1),[Num]);
LOAD
       TransLineID,
       TransID,
       "Num",
       Dim1,
       Dim2,
       Dim3,
       Expression1,
       Expression2,
       Expression3
FROM [lib://AttachedFiles/transactions.qvd] (qvd);
```

Window는 반올림이나 기본 수치 연산과 같은 일반 함수를 지원합니다. 예:

```
Load *, Round(Window(Sum(Salary),Department)) as SumSalary
Load *, Window(Sum(Salary),Department) + 5 as SumSalary
```

Window 함수에 대한 슬라이딩 창을 정의할 수 있습니다. 현재 행에 **Window** 함수를 적용할 때 사용되는 행 수를 설정합니다. 예를 들어, 창을 이전 3개 행과 이후 3개 행으로 설정할 수 있습니다.

구문:

```
Window (input_expr, [partition1, partition2, ...], [sort_type, [sort_expr]],
[filter_expr], [start_expr,end_expr])
```

반환 데이터 유형: LOAD 문으로 만들어진 결과 테이블에 새 필드가 추가되었습니다.

인수:

인수

인수	설명

input_expr	함수에 의해 계산되고 반환되는 입력 표현식입니다. Median(Salary)와 같은 집계 기반 표현식이어야 합니다. 예:
	`Window(Median(Salary)) as MedianSalary`
	입력은 집계가 적용되지 않은 필드 이름일 수도 있습니다. 이 경우 **Window**는 해당 필드에 **Only()** 함수가 적용되는 것처럼 처리합니다. 예:
	`Window(Salary,Department) as WSalary`
	
	선택적으로 입력 표현식을 사용하여 분할을 정의할 수 있습니다. 분할은 결과가 입력 테이블에 새 열로 추가된다는 점을 제외하면 **group by** 절을 통해 수행되는 그룹화와 동일합니다. 파티셔닝은 입력 테이블의 레코드 수를 줄이지 않습니다. 여러 파티션 필드를 정의할 수 있습니다.
	예:
	`LOAD`
	`Window(Max(Sales), City, 'ASC', OrderDate, Sales > 300)`
	`+ AddMonths(OrderDate,-6) as MAX_Sales_City_Last_6_Mos,`
	`Window(Avg(Sales), City, 'ASC', OrderDate, City = 'Portland')`
	`+ AddMonths(OrderDate,-6) as Avg_Sales_Portland_Last_6_Mos,`
	`Window(Max(Sales), City, 'ASC', OrderDate, Sales > 300)`
	`+ AddMonths(OrderDate,-12) as MAX_Sales_City_Last_12_Mos;`
	`LOAD`
	`City,`
	`Sales,`
	`OrderDate`
	`FROM [lib://AttachedFiles/Sales Data.xlsx]`
	`(ooxml, embedded labels, table is [Sales Data]);`

partition1, partition2	input_expr 뒤에는 원하는 만큼의 파티션을 정의할 수 있습니다. 파티션은 집계를 적용할 조합을 정의하는 필드입니다. 집계는 각 파티션에 별도로 적용됩니다. 예: `Window(Avg(Salary), Unit, Department, Country) as AvgSalary` 위에서 파티션은 *Unit*, *Department* 및 *Country*입니다. 파티션은 필수는 아니지만 필드를 적절하게 창 표시하는 데 필요합니다.

인수	설명
sort_type, [sort_expr]]	선택적으로 정렬 유형과 정렬 표현식을 지정합니다. sort_type은 두 값 중 하나를 가질 수 있습니다. • ASC: 오름차순 정렬. • DESC: 내림차순 정렬. sort_type을 정의하는 경우 정렬 표현식을 정의해야 합니다. 파티션 내 행의 순서를 결정하는 표현식입니다. 예: `Window(RecNo(), Department, 'ASC', Year)` 위 예에서 파티션 내의 결과는 *Year* 필드를 기준으로 오름차순으로 정렬됩니다. *정렬 유형 및 정렬 표현식은 기본적으로 **RecNo** 및 **WRank** 함수에만 필요합니다.*
filter_expr	선택적으로 필터 표현식을 추가합니다. 이는 레코드를 계산에 포함할지 여부를 결정하는 부울 표현식입니다. 이 매개 변수는 완전히 생략할 수 있으며 결과적으로 필터가 없어야 합니다. 예: `Window(avg(Salary), Department, 'ASC', Age, EmployeeID=3 Or EmployeeID=7) as wAvgSalary) as wAvgSalaryIfEmpIs3or7`

[start_expr,end_expr]	선택적으로 슬라이딩 윈도우 기능에 대한 인수를 설정합니다. 슬라이딩 윈도우에는 두 가지 인수가 필요합니다.
	• 표현식 시작: 창에 포함할 현재 행 이전의 행 수입니다.
	• 표현식 끝: 창에 포함할 현재 행 뒤의 행 수입니다.
	예를 들어, 앞의 3개 행, 현재 행, 다음 행을 포함하려는 경우:
	`Window(concat(Text(Salary),'-'), Department, 'ASC', Age, Year>0, -3, 1) as WSalaryDepartment`
	모든 이전 행 또는 모든 후속 행을 표시하려면 **Unbounded()** 함수를 사용할 수 있습니다. 예를 들어, 이전 행, 현재 행 및 다음 행을 모두 포함하려면 다음을 수행합니다.
	`Window(concat(Text(Salary),'-'), Department, 'ASC', Age, Year>0, UNBOUNDED(), 1) as WSlidingSalaryDepartment`
	예를 들어, 현재 행의 세 번째 행과 이후의 모든 행을 포함하려면 다음을 수행합니다.
	`Window(concat(Text(Salary),'-'), Department, 'ASC', Age, Year>0, 3, UNBOUNDED()) as WSlidingSalaryDepartment`

예 - 집계가 포함된 필드 추가

예: 집계가 포함된 필드 추가

로드 스크립트

데이터 로드 편집기에서 새 탭을 만든 후 다음 데이터를 인라인 로드로 로드합니다. 결과를 보려면 Qlik Sense에서 아래 테이블을 만듭니다.

```
Transactions:
Load
*,
Window(Avg(transaction_amount),customer_id) as AvgCustTransaction;

Load * Inline [
transaction_id, transaction_date, transaction_amount, transaction_quantity, customer_id, size,
color_code
3750, 20180830, 23.56, 2, 2038593, L, Red
3751, 20180907, 556.31, 6, 203521, M, Orange
3752, 20180916, 5.75, 1, 5646471, S, Blue
3753, 20180922, 125.00, 7, 3036491, L, Black
3754, 20180922, 484.21, 13, 049681, XS, Red
3756, 20180922, 59.18, 2, 2038593, M, Blue
3757, 20180923, 177.42, 21, 203521, XL, Black
3758, 20180924, 153.42, 14, 2038593, L, Red
3759, 20180925, 7.42, 5, 203521, M, Orange
3760, 20180925, 80.12, 18, 5646471, M, Blue
3761, 20180926, 3.42, 7, 3036491, XS, Black
3763, 20180926, 63.55, 12, 049681, S, Red
3763, 20180927, 177.56, 10, 2038593, L, Blue
3764, 20180927, 325.95, 8, 203521, XL, Black
];
```

결과

집계가 포함된 필드를 추가한 결과

transaction_id	transaction_date	transaction_amount	transaction_quantity	customer_id	크기	color_code	AvgCustTransaction
3750	20180830	23.56	2	2038593	L	빨강	103.43
3751	20180907	556.31	6	203521	M	주황색	266.775
3752	20180916	5.75	1	5646471	S	파랑	42.935
3753	20180922	125.00	7	3036491	L	Black	64.21
3754	20180922	484.21	13	049681	XS	빨강	273.88
3756	20180922	59.18	2	2038593	M	파랑	103.43

transaction_id	transaction_date	transaction_amount	transaction_quantity	customer_id	크기	color_code	AvgCustTransaction
3757	20180923	177.42	21	203521	XL	Black	266.775
3758	20180924	153.42	14	2038593	L	빨강	103.43
3759	20180925	7.42	5	203521	M	주황색	266.775
3760	20180925	80.12	18	5646471	M	파랑	42.935
3761	20180926	3.42	7	3036491	XS	Black	64.21
3763	20180926	63.55	12	049681	S	빨강	273.88
3763	20180927	177.56	10	2038593	L	파랑	103.43
3764	20180927	325.95	8	203521	XL	Black	266.775

예 - 특정 값으로 필터링된 집계가 포함된 필드 추가

예: 특정 값으로 필터링된 집계가 포함된 필드 추가

로드 스크립트

데이터 로드 편집기에서 새 탭을 만든 후 다음 데이터를 인라인 로드로 로드합니다. 결과를 보려면 Qlik Sense에서 아래 테이블을 만듭니다.

```
Transactions:
Load
*,
Window(Avg(transaction_amount),customer_id, color_code = 'Blue') as AvgCustTransaction;

Load * Inline [
transaction_id, transaction_date, transaction_amount, transaction_quantity, customer_id, size,
color_code
3750, 20180830, 23.56, 2, 2038593, L, Red
3751, 20180907, 556.31, 6, 203521, M, Orange
3752, 20180916, 5.75, 1, 5646471, S, Blue
3753, 20180922, 125.00, 7, 3036491, L, Black
3754, 20180922, 484.21, 13, 049681, XS, Red
3756, 20180922, 59.18, 2, 2038593, M, Blue
3757, 20180923, 177.42, 21, 203521, XL, Black
3758, 20180924, 153.42, 14, 2038593, L, Red
3759, 20180925, 7.42, 5, 203521, M, Orange
3760, 20180925, 80.12, 18, 5646471, M, Blue
3761, 20180926, 3.42, 7, 3036491, XS, Black
3763, 20180926, 63.55, 12, 049681, S, Red
3763, 20180927, 177.56, 10, 2038593, L, Blue
3764, 20180927, 325.95, 8, 203521, XL, Black
];
```

결과

특정 값으로 필터링된 집계가 포함된 ad 필드를 추가한 결과

transaction_id	transaction_date	transaction_amount	transaction_quantity	customer_id	크기	color_code	AvgCustTransaction
3750	20180830	23.56	2	2038593	L	빨강	-
3751	20180907	556.31	6	203521	M	주황색	-
3752	20180916	5.75	1	5646471	S	파랑	42.94
3753	20180922	125.00	7	3036491	L	Black	-
3754	20180922	484.21	13	049681	XS	빨강	-
3756	20180922	59.18	2	2038593	M	파랑	118.4
3757	20180923	177.42	21	203521	XL	Black	-
3758	20180924	153.42	14	2038593	L	빨강	-
3759	20180925	7.42	5	203521	M	주황색	-
3760	20180925	80.12	18	5646471	M	파랑	42.94
3761	20180926	3.42	7	3036491	XS	Black	-
3763	20180926	63.55	12	049681	S	빨강	-
3763	20180927	177.56	10	2038593	L	파랑	118.4
3764	20180927	325.95	8	203521	XL	Black	-

예 - 슬라이딩 윈도우를 사용하여 필드 추가

예: 슬라이딩 윈도우를 사용하여 필드 추가

로드 스크립트

데이터 로드 편집기에서 새 탭을 만든 후 다음 데이터를 인라인 로드로 로드합니다. 결과를 보려면 Qlik Sense에서 아래 테이블을 만듭니다.

```
Transactions:
Load
*,
Window(Avg(transaction_amount),customer_id, 'ASC', -1, 1, 0, 1) as AvgCustTransaction;

Load * Inline [
transaction_id, transaction_date, transaction_amount, transaction_quantity, customer_id, size,
color_code
3750, 20180830, 23.56, 2, 2038593, L, Red
3751, 20180907, 556.31, 6, 203521, M, Orange
```

```
3752, 20180916, 5.75, 1, 5646471, S, Blue
3753, 20180922, 125.00, 7, 3036491, L, Black
3754, 20180922, 484.21, 13, 049681, XS, Red
3756, 20180922, 59.18, 2, 2038593, M, Blue
3757, 20180923, 177.42, 21, 203521, XL, Black
3758, 20180924, 153.42, 14, 2038593, L, Red
3759, 20180925, 7.42, 5, 203521, M, Orange
3760, 20180925, 80.12, 18, 5646471, M, Blue
3761, 20180926, 3.42, 7, 3036491, XS, Black
3763, 20180926, 63.55, 12, 049681, S, Red
3763, 20180927, 177.56, 10, 2038593, L, Blue
3764, 20180927, 325.95, 8, 203521, XL, Black
];
```

결과

특정 값으로 필터링된 집계가 포함된 ad 필드를 추가한 결과

transaction_id	transaction_date	transaction_amount	transaction_quantity	customer_id	크기	color_code	AvgCustTransaction
3750	20180830	23.56	2	2038593	L	빨강	41.37
3751	20180907	556.31	6	203521	M	주황색	366.865
3752	20180916	5.75	1	5646471	S	파랑	42.935
3753	20180922	125.00	7	3036491	L	Black	64.21
3754	20180922	484.21	13	049681	XS	빨강	273.88
3756	20180922	59.18	2	2038593	M	파랑	106.3
3757	20180923	177.42	21	203521	XL	Black	92.42
3758	20180924	153.42	14	2038593	L	빨강	165.49
3759	20180925	7.42	5	203521	M	주황색	166.685
3760	20180925	80.12	18	5646471	M	파랑	80.12
3761	20180926	3.42	7	3036491	XS	Black	3.42
3763	20180926	63.55	12	049681	S	빨강	177.56
3763	20180927	177.56	10	2038593	L	파랑	63.55
3764	20180927	325.95	8	203521	XL	Black	325.95

제한 사항

Window 다음과 같은 제한 사항이 있습니다.

- **Window**는 특히 메모리 소비 측면에서 리소스를 많이 사용하는 함수입니다.
- **Window**은 Qlik Sense Mobile에서 지원하지 않습니다.

- 차트 표현식은 **Window**을 지원하지 않습니다.
- 다른 **Window** 함수 내에 **Window** 함수를 중첩할 수 없습니다.
- **Window**는 집계 함수 내에서 사용할 수 없습니다.
- **Window**는 전체 테이블을 검색할 수 있어야 합니다.
- **WRank()**, **RecNo()** 및 **RowNo()**는 슬라이딩 창 기능을 사용할 때 **Window**와 함께 사용할 수 없습니다.

WRank - 스크립트 함수:

WRank()는 로드 스크립트에서 테이블 행을 평가하고 각 행에 대해 로드 스크립트에서 평가된 필드 값의 상대적 위치를 표시합니다. 테이블을 평가할 때 함수는 결과를 현재 파티션이 포함된 다른 행의 결과와 비교하고 세그먼트 내 현재 행의 순위를 반환합니다.

테이블의 파티션

Region	Country	Population	Rank(Population)
Americas	Mexico	128,932,753	2
Americas	Canada	37,742,154	3
Americas	United States of America	331,002,651	1
Europe	Sweden	10,099,265	4
Europe	United Kingdom	67,886,011	2
Europe	France	65,273,511	3
Europe	Germany	83,783,942	1

WRank는 **Window** 함수에서만 사용할 수 있습니다. **Window** 함수에는 정렬 유형과 정렬 표현식이 포함되어야 합니다. 순위는 정렬 표현식에 적용됩니다.

구문:

```
WRank ([mode[, fmt]])
```

반환 데이터 유형: dual

인수:

인수

인수	설명
mode	선택적으로 함수 결과의 숫자 표현을 지정합니다.
fmt	선택적으로 함수 결과의 텍스트 표현을 지정합니다.
TOTAL	테이블이 1차원이거나 스크립트 앞에 **TOTAL** 한정자가 있는 경우 이 함수는 전체 열에 따라 평가됩니다. 테이블 또는 테이블과 동등한 것에 여러 세로 차원이 있을 경우 현재 파티션에는 필드 간 정렬 순서에서 마지막 차원이 표시되는 열을 제외하고 모든 차원 열 내의 현재 행과 같은 값을 가진 행만 포함됩니다.

순위는 이중 값으로 반환되며, 각 행은 1과 현재 파티션의 행 수 사이의 정수인 고유 순위를 갖습니다.

여러 행이 동일한 순위를 공유할 경우 **mode** 및 **fmt** 매개 변수를 사용하여 텍스트 및 숫자 표현을 제어할 수 있습니다.

mode
첫 번째 인수인 **mode**는 다음 값을 사용할 수 있습니다.

mode 값

Value	설명
0(기본값)	공유 그룹 내의 모든 순위가 전체 순위의 중간 값에서 낮은 쪽에 속할 경우 공유 그룹 내의 모든 행에 최저 순위가 부여됩니다. 공유 그룹 내의 모든 순위가 전체 순위의 중간 값에서 높은 쪽에 속할 경우 공유 그룹 내의 모든 행에 최고 순위가 부여됩니다. 공유 그룹 내의 순위가 전체 순위의 중간 값에 분포할 경우 모든 행에 전체 파티션의 최고 및 최저 순위의 평균에 해당하는 값이 부여됩니다.
1	모든 행에 최저 순위를 부여합니다.
2	모든 행에 평균 순위를 부여합니다.
3	모든 행에 최고 순위를 부여합니다.
4	첫 번째 행에 최저 순위를 부여하고, 각 행마다 순위가 하나씩 올라갑니다.

fmt

두 번째 인수인 **fmt**에는 다음 값을 적용할 수 있습니다.

fmt 값

Value	설명
0(기본값)	모든 행에 낮은 값 **-** 높은 값을 부여합니다(예: 3 - 4).
1	모든 행에 낮은 값을 부여합니다.
2	첫 번째 행에 낮은 값을 부여하고 나머지 행은 공백으로 채웁니다.

mode 4 및 **fmt** 2의 행 순서는 테이블 필드의 로드 순서에 따라 결정됩니다.

예 - 순위 필드 추가

예: 순위 필드 추가

로드 스크립트

데이터 로드 편집기에서 새 탭을 만든 후 다음 데이터를 인라인 로드로 로드합니다. 결과를 보려면 Qlik Sense에서 아래 테이블을 만듭니다.

```
Transactions:
Load
*,
Window(WRank(0),customer_id, 'Desc', transaction_amount) as TransactionRanking;

Load * Inline [
transaction_id, transaction_date, transaction_amount, transaction_quantity, customer_id, size,
color_code
3750, 20180830, 23.56, 2, 2038593, L, Red
3751, 20180907, 556.31, 6, 203521, M, Orange
```

```
3752, 20180916, 5.75, 1, 5646471, S, Blue
3753, 20180922, 125.00, 7, 3036491, L, Black
3754, 20180922, 484.21, 13, 049681, XS, Red
3756, 20180922, 59.18, 2, 2038593, M, Blue
3757, 20180923, 177.42, 21, 203521, XL, Black
3758, 20180924, 153.42, 14, 2038593, L, Red
3759, 20180925, 7.42, 5, 203521, M, Orange
3760, 20180925, 80.12, 18, 5646471, M, Blue
3761, 20180926, 3.42, 7, 3036491, XS, Black
3763, 20180926, 63.55, 12, 049681, S, Red
3763, 20180927, 177.56, 10, 2038593, L, Blue
3764, 20180927, 325.95, 8, 203521, XL, Black
];
```

결과

순위 필드 추가 결과

transaction_id	transaction_date	transaction_amount	transaction_quantity	customer_id	크기	color_code	TransactionRanking
3750	20180830	23.56	2	2038593	L	빨강	4-4
3751	20180907	556.31	6	203521	M	주황색	1-1
3752	20180916	5.75	1	5646471	S	Blue	2-2
3754	20180922	484.21	13	049681	XS	빨강	1-1
3756	20180922	59.18	2	2038593	M	Blue	3-3
3753	20180922	125.00	7	3036491	L	Black	1-1
3757	20180923	177.42	21	203521	XL	Black	3-3
3758	20180924	153.42	14	2038593	L	빨강	2-2
3759	20180925	7.42	5	203521	M	주황색	4-4
3760	20180925	80.12	18	5646471	M	Blue	1-1
3763	20180926	63.55	12	049681	S	빨강	2-2
3761	20180926	3.42	7	3036491	XS	Black	2-2
3764	20180927	325.95	8	203521	XL	Black	2-2
3763	20180927	177.56	10	2038593	L	Blue	1-1

예 - 한 자리 결과에 대해 fmt를 사용하여 순위 필드 추가

예: 한 자리 결과에 대해 fmt를 사용하여 순위 필드 추가

로드 스크립트

데이터 로드 편집기에서 새 탭을 만든 후 다음 데이터를 인라인 로드로 로드합니다. 결과를 보려면 Qlik Sense에서 아래 테이블을 만듭니다.

```
Transactions:
Load
*,Window(WRank(0,1),customer_id, 'Desc', transaction_amount) as TransactionRanking;

Load * Inline [
transaction_id, transaction_date, transaction_amount, transaction_quantity, customer_id, size,
color_code
3750, 20180830, 23.56, 2, 2038593, L, Red
3751, 20180907, 556.31, 6, 203521, M, Orange
3752, 20180916, 5.75, 1, 5646471, S, Blue
3753, 20180922, 125.00, 7, 3036491, L, Black
3754, 20180922, 484.21, 13, 049681, XS, Red
3756, 20180922, 59.18, 2, 2038593, M, Blue
3757, 20180923, 177.42, 21, 203521, XL, Black
3758, 20180924, 153.42, 14, 2038593, L, Red
3759, 20180925, 7.42, 5, 203521, M, Orange
3760, 20180925, 80.12, 18, 5646471, M, Blue
3761, 20180926, 3.42, 7, 3036491, XS, Black
3763, 20180926, 63.55, 12, 049681, S, Red
3763, 20180927, 177.56, 10, 2038593, L, Blue
3764, 20180927, 325.95, 8, 203521, XL, Black
];
```

결과

한 자리 결과에 대해 fmt를 사용하여 순위 필드를 추가한 결과

transaction_id	transaction_date	transaction_amount	transaction_quantity	customer_id	크기	color_code	TransactionRanking
3750	20180830	23.56	2	2038593	L	빨강	4
3751	20180907	556.31	6	203521	M	주황색	1
3752	20180916	5.75	1	5646471	S	Blue	2
3754	20180922	484.21	13	049681	XS	빨강	1
3756	20180922	59.18	2	2038593	M	Blue	3
3753	20180922	125.00	7	3036491	L	Black	1
3757	20180923	177.42	21	203521	XL	Black	3

transaction_id	transaction_date	transaction_amount	transaction_quantity	customer_id	크기	color_code	TransactionRanking
3758	20180924	153.42	14	2038593	L	빨강	2
3759	20180925	7.42	5	203521	M	주황색	4
3760	20180925	80.12	18	5646471	M	Blue	1
3763	20180926	63.55	12	049681	S	빨강	2
3761	20180926	3.42	7	3036491	XS	Black	2
3764	20180927	325.95	8	203521	XL	Black	2
3763	20180927	177.56	10	2038593	L	Blue	1

예 - 여러 파티션이 있는 순위 필드 추가

예: 여러 파티션이 있는 순위 필드 추가

로드 스크립트

데이터 로드 편집기에서 새 탭을 만든 후 다음 데이터를 인라인 로드로 로드합니다. 결과를 보려면 Qlik Sense에서 아래 테이블을 만듭니다.

```
Transactions:
Load
*,Window(WRank(0,1),customer_id, size, color_code, 'Desc', transaction_amount) as
TransactionRanking;

Load * Inline [
transaction_id, transaction_date, transaction_amount, transaction_quantity, customer_id, size,
color_code
3750, 20180830, 23.56, 2, 2038593, L, Red
3751, 20180907, 556.31, 6, 203521, M, Orange
3752, 20180916, 5.75, 1, 5646471, S, Blue
3753, 20180922, 125.00, 7, 3036491, L, Black
3754, 20180922, 484.21, 13, 049681, XS, Red
3756, 20180922, 59.18, 2, 2038593, M, Blue
3757, 20180923, 177.42, 21, 203521, XL, Black
3758, 20180924, 153.42, 14, 2038593, L, Red
3759, 20180925, 7.42, 5, 203521, M, Orange
3760, 20180925, 80.12, 18, 5646471, M, Blue
3761, 20180926, 3.42, 7, 3036491, XS, Black
3763, 20180926, 63.55, 12, 049681, S, Red
3763, 20180927, 177.56, 10, 2038593, L, Blue
3764, 20180927, 325.95, 8, 203521, XL, Black
];
```

결과

한 자리 결과에 대해 fmt를 사용하여 순위 필드를 추가한 결과

transaction_id	transaction_date	transaction_amount	transaction_quantity	customer_id	크기	color_code	TransactionRanking
3750	20180830	23.56	2	2038593	L	빨강	2
3751	20180907	556.31	6	203521	M	주황색	1
3752	20180916	5.75	1	5646471	S	Blue	1
3754	20180922	484.21	13	049681	XS	빨강	1
3756	20180922	59.18	2	2038593	M	Blue	1
3753	20180922	125.00	7	3036491	L	Black	1
3757	20180923	177.42	21	203521	XL	Black	2
3758	20180924	153.42	14	2038593	L	빨강	1
3759	20180925	7.42	5	203521	M	주황색	2
3760	20180925	80.12	18	5646471	M	Blue	1
3763	20180926	63.55	12	049681	S	빨강	1
3761	20180926	3.42	7	3036491	XS	Black	1
3764	20180927	325.95	8	203521	XL	Black	1
3763	20180927	177.56	10	2038593	L	Blue	1

제한 사항

WRank에는 다음과 같은 제한 사항이 있습니다.

- fmt 값이 0이고 **WRank**에 대한 이중 결과의 텍스트 부분을 사용하려면 **Window(WRank)**와 함께 **Text()**을 사용해야 합니다. 합니다(예: `Text(Window(WRank(0), Unit, 'DESC', Age)) as UnitWRankedByAgeText`).

9 파일 시스템 액세스 제한

보안상의 이유로 표준 모드에서 Qlik Sense는 데이터 로드 스크립트 또는 함수 및 변수에서 파일 시스템이 노출되는 경로를 지원하지 않습니다.

하지만 QlikView에서는 파일 시스템 경로가 지원되므로 QlikView 로드 스크립트를 재사용하기 위해 표준 모드를 비활성화하고 레거시 모드를 사용할 수 있습니다.

표준 모드를 비활성화하면 파일 시스템이 노출되어 보안 위험이 발생할 수 있습니다.

표준 모드 비활성화 (page 1457)

9.1 파일 기반 ODBC 및 OLE DB 데이터 연결에 연결하는 경우의 보안 측면

파일 기반 드라이버를 사용하는 ODBC 및 OLE DB 데이터 연결은 연결 문자열에 연결된 데이터 파일의 경로를 노출합니다. 연결을 편집할 때 데이터 선택 대화 상자 또는 특정 SQL 쿼리에서 경로가 노출될 수 있습니다. 표준 모드와 레거시 모드 둘 다 해당됩니다.

데이터 파일 경로 노출이 염려되는 경우 가능하면 폴더 데이터 연결을 사용하여 데이터 파일에 연결하는 것이 좋습니다.

9.2 표준 모드의 제한 사항

표준 모드에서 사용할 수 없거나 제한이 있는 문, 변수 및 함수가 여럿 있습니다. 데이터 로드 스크립트에서 지원되지 않는 문을 사용하면 로드 스크립트를 실행할 때 오류가 발생합니다. 오류 메시지는 스크립트 로그 파일에서 찾을 수 있습니다. 지원되지 않는 변수 및 함수를 사용해도 오류 메시지나 로드 파일 항목은 생성되지 않습니다. 대신, 함수가 NULL을 반환합니다.

데이터 로드 스크립트를 편집할 때 변수, 문 또는 함수가 지원되지 않는다고 표시되지 않습니다.

시스템 변수

시스템 변수

변수	표준 모드	레거시 모드	정의
Floppy	지원되지 않음	지원됨	첫 번째로 찾은 플로피 드라이브의 드라이브 문자를 반환하며 일반적으로 *a:*입니다.

변수	표준 모드	레거시 모드	정의
CD	지원되지 않음	지원됨	첫 번째로 찾은 CD-ROM 드라이브의 드라이브 문자를 반환합니다. CD-ROM이 발견되지 않으면 *c:*가 반환됩니다.
QvPath	지원되지 않음	지원됨	Qlik Sense 실행 파일에 대한 찾아보기 문자열을 반환합니다.
QvRoot	지원되지 않음	지원됨	Qlik Sense 실행 파일의 루트 디렉터리를 반환합니다.
QvWorkPath	지원되지 않음	지원됨	현재 Qlik Sense 앱에 대한 탐색 문자열을 반환합니다.
QvWorkRoot	지원되지 않음	지원됨	현재 Qlik Sense 앱의 루트 디렉터리를 반환합니다.
WinPath	지원되지 않음	지원됨	Windows에 대한 탐색 문자열을 반환합니다.
WinRoot	지원되지 않음	지원됨	Windows의 루트 디렉터리를 반환합니다.
$(include=...)	지원되는 입력: 라이브러리 연결을 사용하는 경로	지원되는 입력: 라이브러리 연결 또는 파일 시스템을 사용하는 경로	**Include/Must_Include** 변수는 스크립트에 포함되어 스크립트 코드로 평가되어야 하는 텍스트를 포함한 파일을 지정합니다. 데이터를 추가하는 데 사용되지 않습니다. 스크립트 코드의 일부를 별도의 텍스트 파일에 저장한 후 여러 앱에서 재사용할 수 있습니다. 이 변수는 사용자 정의 변수입니다.

정규 스크립트 문

정규 스크립트 문

문	표준 모드	레거시 모드	정의
Binary	지원되는 입력: 라이브러리 연결을 사용하는 경로	지원되는 입력: 라이브러리 연결 또는 파일 시스템을 사용하는 경로	**binary** 문은 다른 앱에서 데이터를 로드하는 데 사용됩니다.
Connect	지원되는 입력: 라이브러리 연결을 사용하는 경로	지원되는 입력: 라이브러리 연결 또는 파일 시스템을 사용하는 경로	**CONNECT** 문은 OLE DB/ODBC 인터페이스를 통해 일반 데이터베이스에 대한 Qlik Sense 액세스를 정의하는 데 사용됩니다. ODBC의 경우, 먼저 ODBC 관리자를 사용하여 데이터 소스를 지정해야 합니다.
Directory	지원되는 입력: 라이브러리 연결을 사용하는 경로	지원되는 입력: 라이브러리 연결 또는 파일 시스템을 사용하는 경로	**Directory** 문은 새 **Directory** 문이 만들어질 때까지 후속 **LOAD** 문에서 데이터 파일을 탐색할 디렉터리를 정의합니다.
Execute	지원되지 않음	지원되는 입력: 라이브러리 연결 또는 파일 시스템을 사용하는 경로	**Execute** 문은 Qlik Sense가 데이터를 로드하는 동안 다른 프로그램을 실행하는 데 사용됩니다. 예를 들어, 필요한 규칙을 만드는 데 사용됩니다.
LOAD from ...	지원되는 입력: 라이브러리 연결을 사용하는 경로	지원되는 입력: 라이브러리 연결 또는 파일 시스템을 사용하는 경로	**LOAD** 문은 파일, 스크립트에 정의된 데이터, 이전에 로드한 테이블, 웹 페이지, 이후 **SELECT** 문의 결과에서 필드를 로드하거나 자동으로 데이터를 생성하여 필드를 로드합니다.
Store into ...	지원되는 입력: 라이브러리 연결을 사용하는 경로	지원되는 입력: 라이브러리 연결 또는 파일 시스템을 사용하는 경로	**Store** 문은 QVD, Parquet, CSV 또는 TXT 파일을 만듭니다.

스크립트 제어 문

스크립트 제어 문

문	표준 모드	레거시 모드	정의
For each... filelist mask/dirlist mask	지원되는 입력: 라이브러리 연결을 사용하는 경로 반환되는 출력: 라이브러리 연결	지원되는 입력: 라이브러리 연결 또는 파일 시스템을 사용하는 경로 반환되는 출력: 입력에 따라, 라이브러리 연결 또는 파일 시스템 경로	filelist mask 구문을 사용하면 현재 디렉터리에서 **filelist mask**와 일치하는 모든 파일의 쉼표로 구분된 목록이 생성됩니다. **dirlist mask** 구문을 사용하면 현재 디렉터리에서 디렉터리 이름 마스크와 일치하는 모든 디렉터리의 쉼표로 구분된 목록이 생성됩니다.

파일 함수

파일 함수

함수	표준 모드	레거시 모드	정의
Attribute()	지원되는 입력: 라이브러리 연결을 사용하는 경로	지원되는 입력: 라이브러리 연결 또는 파일 시스템을 사용하는 경로	다양한 미디어 파일의 메타 태그 값을 텍스트로 반환합니다.
ConnectString()	반환되는 출력: 라이브러리 연결 이름	입력에 따라, 라이브러리 연결 이름 또는 실제 연결	ODBC 또는 OLE DB 연결에 대한 활성 연결 문자열을 반환합니다.
FileDir()	반환되는 출력: 라이브러리 연결	반환되는 출력: 입력에 따라, 라이브러리 연결 또는 파일 시스템 경로	**FileDir** 함수는 현재 읽고 있는 테이블 파일의 디렉터리에 대한 경로가 포함된 문자열을 반환합니다.
FilePath()	반환되는 출력: 라이브러리 연결	반환되는 출력: 입력에 따라, 라이브러리 연결 또는 파일 시스템 경로	**FilePath** 함수는 현재 읽고 있는 테이블 파일에 대한 전체 경로가 포함된 문자열을 반환합니다.

함수	표준 모드	레거시 모드	정의
FileSize()	지원되는 입력: 라이브러리 연결을 사용하는 경로	지원되는 입력: 라이브러리 연결 또는 파일 시스템을 사용하는 경로	**FileSize** 함수는 filename 파일의 바이트 단위 크기 또는 filename이 지정되지 않은 경우 현재 읽고 있는 테이블 파일의 바이트 단위 크기가 포함된 정수를 반환합니다.
FileTime()	지원되는 입력: 라이브러리 연결을 사용하는 경로	지원되는 입력: 라이브러리 연결 또는 파일 시스템을 사용하는 경로	**FileTime** 함수는 지정된 파일의 마지막 수정에 대한 UTC 서식의 타임스탬프를 반환합니다. 파일이 지정되지 않은 경우 함수는 현재 읽은 테이블 파일의 마지막 수정에 대한 타임스탬프(UTC)를 반환합니다.
GetFolderPath()	지원되지 않음	반환되는 출력: 절대 경로	**GetFolderPath** 함수는 Microsoft Windows *SHGetFolderPath* 함수의 값을 반환합니다. 이 함수는 Microsoft Windows 폴더의 이름을 입력으로 사용하여 이 폴더의 전체 경로를 반환합니다.
QvdCreateTime()	지원되는 입력: 라이브러리 연결을 사용하는 경로	지원되는 입력: 라이브러리 연결 또는 파일 시스템을 사용하는 경로	이 스크립트 함수는 QVD 파일(있는 경우)의 XML 헤더 타임스탬프를 반환하며, 그렇지 않으면 NULL을 반환합니다. 타임스탬프에서 시간은 UTC로 제공됩니다.
QvdFieldName()	지원되는 입력: 라이브러리 연결을 사용하는 경로	지원되는 입력: 라이브러리 연결 또는 파일 시스템을 사용하는 경로	이 스크립트 함수는 QVD 파일 내의 필드 번호 **fieldno**의 이름을 반환합니다. 필드가 존재하지 않으면 NULL이 반환됩니다.

함수	표준 모드	레거시 모드	정의
QvdNoOfFields()	지원되는 입력: 라이브러리 연결을 사용하는 경로	지원되는 입력: 라이브러리 연결 또는 파일 시스템을 사용하는 경로	이 스크립트 함수는 QVD 파일 내의 필드 수를 반환합니다.
QvdNoOfRecords()	지원되는 입력: 라이브러리 연결을 사용하는 경로	지원되는 입력: 라이브러리 연결 또는 파일 시스템을 사용하는 경로	이 스크립트 함수는 QVD 파일 내의 현재 레코드 수를 반환합니다.
QvdTableName()	지원되는 입력: 라이브러리 연결을 사용하는 경로	지원되는 입력: 라이브러리 연결 또는 파일 시스템을 사용하는 경로	이 스크립트 함수는 QVD 파일에 저장된 테이블의 이름을 반환합니다.

시스템 함수

시스템 함수

함수	표준 모드	레거시 모드	정의
DocumentPath()	지원되지 않음	반환되는 출력: 절대 경로	이 함수는 현재 Qlik Sense 앱의 전체 경로가 포함된 문자열을 반환합니다.
GetRegistryString()	지원되지 않음	지원됨	지정된 레지스트리 경로를 가진 명명된 레지스트리 키 값을 반환합니다. 이 함수는 차트 및 스크립트 등에서 사용할 수 있습니다.

9.3 표준 모드 비활성화

라이브러리 연결뿐만 아니라 절대 또는 상대 파일 경로를 참조하는 QlikView 로드 스크립트를 재사용하기 위해 표준 모드를 비활성화(다른 말로는 레거시 모드 설정)할 수 있습니다.

표준 모드를 비활성화하면 파일 시스템이 노출되어 보안 위험이 발생할 수 있습니다.

Qlik Sense

Qlik Sense에서는 QMC에서 **표준 모드** 속성을 사용하여 표준 모드를 비활성화할 수 있습니다.

Qlik Sense Desktop

Qlik Sense Desktop에서는 *Settings.ini*에서 표준/레거시 모드를 설정할 수 있습니다.

기본 설치 위치를 사용하여 Qlik Sense Desktop를 설치한 경우 *Settings.ini*는 *C:\Users\{user}\Documents\Qlik\Sense\Settings.ini*에 있습니다. 선택한 폴더에 Qlik Sense Desktop를 설치한 경우 *Settings.ini*는 해당 설치 경로의 *Engine* 폴더에 있습니다.

다음과 같이 하십시오.

1. 텍스트 편집기에서 *Settings.ini*를 엽니다.
2. *StandardReload=1*을 *StandardReload=0*으로 변경합니다.
3. 파일을 저장하고 Qlik Sense Desktop를 시작합니다.

이제 Qlik Sense Desktop가 레거시 모드로 실행됩니다.

설정

StandardReload에 대해 사용 가능한 설정은 다음과 같습니다.

- 1(표준 모드)
- 0(레거시 모드)

10 차트 수준 스크립팅

차트 데이터를 수정할 때 여러 문으로 구성된 Qlik Sense 스크립트의 하위 집합을 사용합니다. 하나의 문은 정규 스크립트 문 또는 스크립트 제어 문일 수 있습니다. 특정 문은 접두사가 선행될 수 있습니다.

정규 문은 일반적으로 어떤 방식으로든 데이터를 편집하는 데 사용됩니다. 이러한 문은 스크립트에서 줄 수에 관계없이 작성할 수 있으며, 항상 세미콜론(";")으로 종결되어야 합니다.

제어 문은 일반적으로 스크립트 실행 흐름을 제어하는 데 사용됩니다. 제어 문의 각 절은 하나의 스크립트 줄 안에 유지되어야 하며 세미콜론 또는 줄의 끝으로 종결될 수 있습니다.

접두사는 해당되는 정규 문에는 적용할 수 있지만 제어 문에는 적용할 수 없습니다.

모든 스크립트 키워드는 소문자와 대문자를 원하는 대로 조합하여 입력할 수 있습니다. 하지만 문에 사용되는 필드 및 변수 이름은 대/소문자가 구분됩니다.

이 섹션에서는 차트 데이터를 수정할 때 사용되는 스크립트의 하위 집합에서 사용할 수 있는 모든 스크립트 문, 제어 문 및 접두사의 사전순 목록을 찾을 수 있습니다.

10.1 제어 문

차트 데이터를 수정할 때 여러 문으로 구성된 Qlik Sense 스크립트의 하위 집합을 사용합니다. 하나의 문은 정규 스크립트 문 또는 스크립트 제어 문일 수 있습니다.

제어 문은 일반적으로 스크립트 실행 흐름을 제어하는 데 사용됩니다. 제어 문의 각 절은 하나의 스크립트 줄 안에 유지되어야 하며 세미콜론 또는 줄의 끝으로 종결될 수 있습니다.

접두사는 제어 문에 적용되지 않습니다.

모든 스크립트 키워드는 소문자와 대문자를 원하는 대로 조합하여 입력할 수 있습니다.

차트 수정자 제어 문 개요

각 함수는 개요가 끝난 후에 더 자세히 설명합니다. 구문에서 함수 이름을 클릭하여 해당 함수에 대한 상세 설명에 즉시 액세스할 수도 있습니다.

Call
call 제어 문은 앞에 **sub** 문으로 정의된 서브루틴을 호출합니다.

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

Do..loop
do..loop 제어 문은 논리 조건이 충족될 때까지 하나 또는 여러 문을 실행하는 스크립트 반복 구조입니다.

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

End
End 스크립트 키워드는 **If**, **Sub** 및 **Switch** 절을 닫는 데 사용됩니다.

Exit

Exit 스크립트 키워드는 **Exit Script** 문에 속하지만 **Do**, **For** 또는 **Sub** 절을 종료할 때도 사용할 수 있습니다.

Exit script

이 제어 문은 스크립트 실행을 중지합니다. 스크립트 어느 곳에나 삽입할 수 있습니다.

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

For..next

for..next 제어 문은 카운터를 사용하는 스크립트 반복 구조입니다. 지정된 하한 및 상한 사이의 각 카운터 변수 값에 대해 **for**와 **next**로 묶인 루프 내의 문이 실행됩니다.

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

[statements]

[exit for [ ( when | unless ) condition ]

[statements]

Next [counter]
```

For each ..next

for each..next 제어 문은 쉼표로 구분된 목록의 각 값을 대상으로 하나 또는 여러 문을 실행하는 스크립트 반복 구조입니다. 목록의 각 값에 대해 **for**와 **next**로 묶인 루프 내의 문이 실행됩니다.

```
For each..next var in list

[statements]

[exit for [ ( when | unless ) condition ]

[statements]

next [var]
```

If..then

if..then 제어 문은 하나 또는 여러 논리 조건에 따라 스크립트 실행을 다른 경로로 전환하는 스크립트 선택 구조입니다.

> ***if..then** 문은 제어 문이므로 세미콜론이나 줄 끝(EOL)으로 끝나며 여기에 포함될 수 있는 4개의 절(**if..then**, **elseif..then**, **else** 및 **end if**)은 줄 경계를 넘지 않아야 합니다.*

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

  [ statements ]

{ elseif condition then
```

```
  [ statements ] }

[ else

  [ statements ] ]

end if
```

Next
Next 스크립트 키워드는 **For** 루프를 닫는 데 사용됩니다.

Sub
sub..end sub 제어 문은 **call** 문에서 호출할 수 있는 서브루틴을 정의합니다.

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

Switch
switch 제어 문은 표현식의 값에 따라 스크립트 실행을 다른 경로로 전환하는 스크립트 선택 구조입니다.

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

To
To 스크립트 키워드는 여러 스크립트 문에서 사용됩니다.

Call

call 제어 문은 앞에 **sub** 문으로 정의된 서브루틴을 호출합니다.

구문:

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

인수:

인수

인수	설명
name	서브루틴의 이름입니다.
paramlist	서브루틴으로 전달할 실제 매개 변수의 쉼표로 구분된 목록입니다. 목록의 각 항목은 필드 이름, 변수 또는 임의의 표현식이 될 수 있습니다.

call 문으로 호출하는 서브루틴은 스크립트 실행 시 초반에 발견되는 **sub** 문으로 정의되어 있어야 합니다.

매개 변수는 서브루틴으로 복사되며, **call** 문에 있는 매개 변수가 변수이고 표현식이 아니라면 서브루틴이 종료될 때 다시 역으로 복사됩니다.

제한 사항:

- **call** 문은 제어 문이고 세미콜론 또는 줄 끝 중 하나로 끝나므로 줄 경계를 넘어가지 말아야 합니다.
- 제어 문(예: `if..then`) 내에서 `Sub..end sub`를 사용하여 서브루틴을 정의하면 동일한 제어 문 내에서만 서브루틴을 호출할 수 있습니다.

Do..loop

do..loop 제어 문은 논리 조건이 충족될 때까지 하나 또는 여러 문을 실행하는 스크립트 반복 구조입니다.

구문:

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

> ***do..loop** 문은 제어 문이므로 세미콜론이나 줄 끝(EOL)으로 끝나며 여기에 포함될 수 있는 세 절 (**do**,**exit do** 및 **loop**)은 줄 경계를 넘지 않아야 합니다.*

인수:

인수

인수	설명
condition	True 또는 False로 평가되는 논리 표현식입니다.
statements	하나 이상의 Qlik Sense 스크립트 문의 그룹입니다.
while / until	**while** 또는 **until** 조건절은 **do..loop** 문에서 한 번만 나와야 합니다. 즉, **do** 다음이나 **loop** 다음에 나올 수 있습니다. 각 조건은 처음 발견될 때만 해석되지만 루프 내에서는 발견될 때마다 평가됩니다.
exit do	루프 내에서 **exit do** 절이 나올 경우 스크립트 실행이 루프의 끝을 나타내는 **loop** 절 다음 첫 번째 문으로 전환됩니다. 선택적으로 **when** 또는 **unless** 접미사를 사용하여 **exit do** 절을 조건부로 만들 수 있습니다.

End

End 스크립트 키워드는 **If**, **Sub** 및 **Switch** 절을 닫는 데 사용됩니다.

Exit

Exit 스크립트 키워드는 **Exit Script** 문에 속하지만 **Do**, **For** 또는 **Sub** 절을 종료할 때도 사용할 수 있습니다.

Exit script

이 제어 문은 스크립트 실행을 중지합니다. 스크립트 어느 곳에나 삽입할 수 있습니다.

구문:

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

exit script 문은 제어 문이고 세미콜론 또는 줄 끝 중 하나로 끝나므로 줄 경계를 넘어가지 말아야 합니다.

인수:

인수

인수	설명
condition	True 또는 False로 평가되는 논리 표현식입니다.
when / unless	선택적으로 **when** 또는 **unless** 절을 사용하여 **exit script** 문을 조건부로 만들 수 있습니다.

```
//Exit script
Exit Script;

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

For..next

for..next 제어 문은 카운터를 사용하는 스크립트 반복 구조입니다. 지정된 하한 및 상한 사이의 각 카운터 변수 값에 대해 **for**와 **next**로 묶인 루프 내의 문이 실행됩니다.

구문:

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

[statements]

[exit for [ ( when | unless ) condition ]

[statements]

Next [counter]
```

expr1, *expr2* 및 *expr3* 표현식은 루프에 처음 진입할 때만 평가됩니다. counter 변수의 값은 루프 내의 문에서 변경할 수 있지만 이는 좋은 프로그래밍 방법이 아닙니다.

루프 내에서 **exit for** 절이 나올 경우 스크립트 실행이 루프의 끝을 나타내는 **next** 절 다음 첫 번째 문으로 전환됩니다. 선택적으로 **when** 또는 **unless** 접미사를 사용하여 **exit for** 절을 조건부로 만들 수 있습니다.

for..next 문은 제어 문이므로 세미콜론이나 줄 끝(EOL)으로 끝나며 여기에 포함될 수 있는 세 절 (for..to..step,exit for 및 next)은 줄 경계를 넘지 않아야 합니다.

인수:

인수

인수	설명
counter	변수 이름입니다. *counter*가 **next** 다음에 지정된 경우, 이 이름은 해당하는 **for** 다음에 오는 이름과 동일해야 합니다.
expr1	루프를 실행할 *counter* 변수의 첫 번째 값을 결정하는 표현식입니다.
expr2	루프를 실행할 *counter* 변수의 마지막 값을 결정하는 표현식입니다.
expr3	루프가 실행될 때마다 *counter* 변수의 증가분을 나타내는 값을 결정하는 표현식입니다.
condition	True 또는 False로 평가되는 논리 표현식입니다.
statements	하나 이상의 Qlik Sense 스크립트 문의 그룹입니다.

For each..next

for each..next 제어 문은 쉼표로 구분된 목록의 각 값을 대상으로 하나 또는 여러 문을 실행하는 스크립트 반복 구조입니다. 목록의 각 값에 대해 **for**와 **next**로 묶인 루프 내의 문이 실행됩니다.

구문:
특수한 구문을 사용하면 현재 디렉터리 내의 파일과 디렉터리 이름으로 목록을 생성할 수 있습니다.

```
for each var in list

[statements]

[exit for [ ( when | unless ) condition ]

[statements]

next [var]
```

인수:

인수

인수	설명
var	각 루프 실행을 위해 목록에서 새 값을 가져오는 스크립트 변수 이름입니다. **var**가 **next** 다음에 지정된 경우, 이 이름은 해당하는 **for each** 다음에 오는 이름과 동일해야 합니다.

var 변수의 값은 루프 내의 문에서 변경할 수 있지만 이것은 좋은 프로그래밍 방법이 아닙니다.

루프 내에서 **exit for** 절이 나올 경우 스크립트 실행이 루프의 끝을 나타내는 **next** 절 다음 첫 번째 문으로 전환됩니다. 선택적으로 **when** 또는 **unless** 접미사를 사용하여 **exit for** 절을 조건부로 만들 수 있습니다.

> ***for each..next** 문은 제어 문이므로 세미콜론이나 줄 끝(EOL)으로 끝나며 여기에 포함될 수 있는 세 절(**for each**,**exit for** 및 **next**)은 줄 경계를 넘지 않아야 합니다.*

구문:

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

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

인수

인수	설명
constant	임의의 숫자 또는 문자열입니다. 스크립트에 직접 포함되는 문자열은 작은따옴표로 묶어야 합니다. 작은따옴표가 없는 문자열은 변수로 해석되며 변수의 값이 사용됩니다. 숫자는 작은따옴표로 묶을 필요가 없습니다.
expression	임의의 표현식입니다.
mask	표준 와일드카드 문자 ***** 및 **?**를 비롯한 유효한 파일 이름 문자를 포함할 수 있는 파일 또는 디렉터리 이름 마스크입니다. 절대 파일 경로 또는 lib:// 경로를 사용할 수 있습니다.
condition	True 또는 False로 평가되는 논리 표현식입니다.
statements	하나 이상의 Qlik Sense 스크립트 문의 그룹입니다.
filelist mask	이 구문을 사용하면 현재 디렉터리에서 파일 이름 마스크와 일치하는 모든 파일의 쉼표로 구분된 목록이 생성됩니다. *이 인수는 표준 모드의 라이브러리 연결만 지원합니다.*
dirlist mask	이 구문을 사용하면 현재 폴더에서 폴더 이름 마스크와 일치하는 모든 폴더의 쉼표로 구분된 목록이 생성됩니다. *이 인수는 표준 모드의 라이브러리 연결만 지원합니다.*
fieldvaluelist mask	이 구문은 Qlik Sense로 이미 로드된 필드 값을 통해 반복합니다.

> *Qlik 웹 저장소 공급자 커넥터 및 기타 DataFiles 연결은 와일드카드(***** 및 **?**) 문자를 사용하는 필터 마스크를 지원하지 않습니다.*

Example 1: 파일 목록 로드

```
// 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: 디스크의 파일 목록 만들기

이 예에서는 폴더에 있는 모든 Qlik Sense 관련 파일 목록을 로드합니다.

```
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: 필드 값 a를 통해 반복

이 예에서는 로드된 FIELD 값의 목록을 통해 반복하고 새 필드인 NEWFIELD를 생성합니다. FIELD의 각 값에 대해서는 두 NEWFIELD 레코드가 생성됩니다.

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

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

결과 테이블은 다음과 같습니다.

Example table

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

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

if..then 제어 문은 하나 또는 여러 논리 조건에 따라 스크립트 실행을 다른 경로로 전환하는 스크립트 선택 구조입니다.

제어 문은 일반적으로 스크립트 실행 흐름을 제어하는 데 사용됩니다. 차트 표현식에서는 **if** 조건부 함수를 대신 사용합니다.

구문:

```
If  condition then

  [ statements ]

{ elseif condition then

  [ statements ] }

[ else

  [ statements ] ]

end if
```

if..then 문은 제어 문이므로 세미콜론이나 줄 끝(EOL)으로 끝나며 여기에 포함될 수 있는 4개의 절(**if..then**, **elseif..then**, **else** 및 **end if**)은 줄 경계를 넘지 않아야 합니다.

인수:

인수

인수	설명
condition	True 또는 False로 평가되는 논리 표현식입니다.
statements	하나 이상의 Qlik Sense 스크립트 문의 그룹입니다.

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

Next 스크립트 키워드는 **For** 루프를 닫는 데 사용됩니다.

Sub..end sub

sub..end sub 제어 문은 **call** 문에서 호출할 수 있는 서브루틴을 정의합니다.

구문:

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

인수는 서브루틴에 복사되며, **call** 문에 있는 해당 실제 매개 변수가 변수 이름일 경우 서브루틴을 종료할 때 다시 복제됩니다.

서브루틴에 **call** 문에 의해 전달된 실제 매개 변수의 수보다 더 많은 공식 매개 변수가 있는 경우 추가 매개 변수는 NULL로 초기화되며 서브루틴 내에서 로컬 변수로 사용될 수 있습니다.

인수:

인수

인수	설명
name	서브루틴의 이름입니다.

인수	설명
paramlist	서브루틴의 공식 매개 변수에 대한 쉼표로 구분된 변수 이름 목록입니다. 이들은 서브루틴 내의 어떠한 변수로도 사용할 수 있습니다.
statements	하나 이상의 Qlik Sense 스크립트 문의 그룹입니다.

제한 사항:

- **sub** 문은 제어 문이므로 세미콜론이나 줄 끝(EOL)으로 끝나며 여기에 포함될 수 있는 두 절(**sub** 및 **end sub**)은 줄 경계를 넘지 않아야 합니다.
- 제어 문(예: `if..then`) 내에서 `Sub..end sub`를 사용하여 서브루틴을 정의하면 동일한 제어 문 내에서만 서브루틴을 호출할 수 있습니다.

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: - 매개 변수 전달

```
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)
```

상기 예의 결과는 서브루틴 내에서 로컬로 이루어지며, A는 1로, B는 4로, C는 NULL로 초기화됩니다.

서브루틴을 종료할 때 전역 변수 A의 값은 2가 됩니다(서브루틴에서 다시 복사됨). 두 번째 실제 매개 변수 "(X+1)*2"는 변수가 아니기 때문에 다시 복사되지 않습니다. 마지막으로 전역 변수 C는 서브루틴 호출에 의해 영향을 받지 않습니다.

Switch..case..default..end switch

switch 제어 문은 표현식의 값에 따라 스크립트 실행을 다른 경로로 전환하는 스크립트 선택 구조입니다.

구문:

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

> ***switch** 문은 제어 문이므로 세미콜론이나 줄 끝(EOL)으로 끝나며 여기에 포함될 수 있는 4개의 절(**switch**, **case**, **default** 및 **end switch**)은 줄 경계를 넘지 않아야 합니다.*

인수:

인수

인수	설명
expression	임의의 표현식입니다.
valuelist	쉼표로 구분된 값 목록이며, 표현식의 값과 비교됩니다. 값 목록의 값이 표현식의 값과 같은 것으로 발견된 첫 번째 그룹의 문을 사용하여 스크립트 실행이 계속됩니다. 값 목록의 각 값은 임의의 표현식일 수 있습니다. **case** 절에서 일치하는 항목이 발견되지 않는 경우 **default** 절 아래의 문이 실행됩니다(지정한 경우).
statements	하나 이상의 Qlik Sense 스크립트 문의 그룹입니다.

```
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

To 스크립트 키워드는 여러 스크립트 문에서 사용됩니다.

10.2 접두사

접두사는 해당되는 정규 문에는 적용할 수 있지만 제어 문에는 적용할 수 없습니다.

모든 스크립트 키워드는 소문자와 대문자를 원하는 대로 조합하여 입력할 수 있습니다. 하지만 문에 사용되는 필드 및 변수 이름은 대/소문자가 구분됩니다.

차트 수정자 접두사 개요

각 함수는 개요가 끝난 후에 더 자세히 설명합니다. 구문에서 함수 이름을 클릭하여 해당 함수에 대한 상세 설명에 즉시 액세스할 수도 있습니다.

Add

Add 접두사를 스크립트의 **LOAD** 또는 **SELECT** 문에 추가하여 레코드를 다른 테이블에 추가하도록 지정할 수 있습니다. 또한 이 문을 부분 로드에서 실행하도록 지정합니다. **Add** 접두사는 **Map** 문에도 사용할 수 있습니다.

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

Replace

Replace 접두사를 스크립트의 **LOAD** 또는 **SELECT** 문에 추가하여 로드된 테이블이 다른 테이블을 대체하도록 지정할 수 있습니다. 또한 이 문을 부분 로드에서 실행하도록 지정합니다. **Replace** 접두사는 **Map** 문에도 사용할 수 있습니다.

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

Add

차트 수정 컨텍스트에서 **Add** 접두사는 **LOAD**와 함께 사용되어 Qlik 연관 엔진으로 계산된 하이퍼큐브를 나타내는 *HC1* 테이블에 값을 추가합니다. 하나 이상의 열을 지정할 수 있습니다. 누락된 값은 Qlik 연관 엔진에 의해 자동으로 채워집니다.

구문:

```
Add loadstatement
```

이 예에서는 인라인 문의 *Dates* 및 *Sales* 열에 두 개의 행을 추가합니다.

```
Add Load
x as Dates,
y as Sales
Inline
[
Dates,Sales
2001/09/1,1000
2001/09/10,-300
]
```

Replace

차트 수정 컨텍스트에서 **Replace** 접두사는 스크립트에서 정의한 계산된 값으로 *HC1* 테이블의 모든 값을 변경합니다.

구문:

```
Replace loadstatement
```

이 예에서는 x 및 y의 합계로 z 열의 모든 값을 덮어씁니다.

```
Replace Load
x+y as z
Resident HC1;
```

10.3 정규 문

정규 문은 일반적으로 어떤 방식으로든 데이터를 편집하는 데 사용됩니다. 이러한 문은 스크립트에서 줄 수에 관계없이 작성할 수 있으며, 항상 세미콜론(";")으로 종결되어야 합니다.

모든 스크립트 키워드는 소문자와 대문자를 원하는 대로 조합하여 입력할 수 있습니다. 하지만 문에 사용되는 필드 및 변수 이름은 대/소문자가 구분됩니다.

차트 수정자 정규 문 개요

각 함수는 개요가 끝난 후에 더 자세히 설명합니다. 구문에서 함수 이름을 클릭하여 해당 함수에 대한 상세 설명에 즉시 액세스할 수도 있습니다.

LOAD

차트 수정 컨텍스트에서 **LOAD** 문은 스크립트에 정의된 데이터 또는 이전에 로드된 테이블에서 하이퍼큐브에 추가 데이터를 로드합니다. 분석 연결에서 데이터를 로드할 수도 있습니다.

***LOAD** 문에는 **Replace** 또는 **Add** 접두사가 있어야 합니다. 그렇지 않으면 거부됩니다.*

```
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
let 문은 스크립트 변수 정의에 사용된다는 점에서 **set** 문과 차이가 있습니다. **let** 문은 **set** 문과는 반대로 변수에 할당하기 전 스크립트 런타임에 '='의 오른쪽에 있는 표현식을 평가합니다.

```
Let variablename=expression
```

Set
set 문은 스크립트 변수를 정의하는 데 사용됩니다. 문자열, 경로, 드라이브 등을 대체하는 데 사용할 수 있습니다.

```
Set variablename=string
```

Put
Put 문은 하이퍼큐브에서 일부 숫자 값을 설정하는 데 사용됩니다.

HCValue
HCValue 문은 지정된 열의 행에서 값을 검색하는 데 사용됩니다.

Load

차트 수정 컨텍스트에서 **LOAD** 문은 스크립트에 정의된 데이터 또는 이전에 로드된 테이블에서 하이퍼큐브에 추가 데이터를 로드합니다. 분석 연결에서 데이터를 로드할 수도 있습니다.

***LOAD** 문에는 **Replace** 또는 **Add** 접두사가 있어야 합니다. 그렇지 않으면 거부됩니다.*

구문:

```
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 ]
```

인수:

인수

인수	설명
fieldlist	*fieldlist ::= (* | field{, * | field })* 로드할 필드의 목록입니다. 필드 목록에 *를 사용하면 테이블의 모든 필드를 지정할 수 있습니다. *field ::= (fieldref | expression) [***as** *aliasname]* 필드 정의는 리터럴, 기존 필드에 대한 참조 또는 표현식을 항상 포함해야 합니다. *fieldref ::= (fieldname |@fieldnumber |@startpos:endpos [**I** | **U** | **R** | **B** | **T**])* *fieldname*은 테이블의 필드 이름과 동일한 텍스트입니다. 필드 이름에 공백 등이 포함된 경우 곧은 큰따옴표 또는 대괄호로 묶어야 합니다. 필드 이름을 명시적으로 사용할 수 없는 경우도 있습니다. 이 경우 다른 표기법이 사용됩니다. *@fieldnumber*는 구분된 테이블 파일의 필드 번호를 나타냅니다. 이 숫자는 "@"가 앞에 오는 양의 정수여야 합니다. 숫자는 항상 1부터 시작하며 필드 개수까지 지정할 수 있습니다. *@startpos*:*endpos*는 고정 길이 레코드를 가진 파일 내 필드의 시작과 끝 위치를 나타냅니다. 위치는 모두 양의 정수여야 합니다. 이 두 숫자는 "@"가 앞에 와야 하며 콜론으로 구분되어야 합니다. 숫자는 항상 1부터 시작하며 위치 개수까지 지정할 수 있습니다. 마지막 필드의 **n**은 종료 위치로 사용됩니다. • *@startpos*:*endpos*의 바로 뒤에 문자 **I** 또는 **U**가 오는 경우 로드된 바이트는 부호가 있는(**I**) 이진수 또는 부호가 없는(**U**) 정수(Intel 바이트 순서)로 해석됩니다. 로드된 위치의 수는 1, 2 또는 4여야 합니다. • *@startpos*:*endpos*의 바로 뒤에 문자 **R**이 오는 경우, 로드된 바이트는 이진 실수(IEEE 32비트 또는 64비트 부동 소수점)로 해석됩니다. 로드된 위치의 수는 4 또는 8이어야 합니다. • *@startpos*:*endpos*의 바로 뒤에 문자 **B**가 오는 경우, 로드된 바이트는 COMP-3 표준에 따라 BCD (Binary Coded Decimal) 숫자로 해석됩니다. 임의의 바이트 수가 지정될 수 있습니다. *expression*은 같은 테이블 내의 하나 또는 여러 다른 필드에 따라 숫자 함수 또는 문자열 함수가 될 수 있습니다. 자세한 내용은 표현식의 구문을 참조하십시오. **as**는 필드에 새 이름을 할당하는 데 사용됩니다.

인수	설명
inline	**inline**은 데이터를 스크립트 내에 입력해야 하고 파일에서 로드되지 않도록 해야 하는 경우에 사용됩니다. *data ::= [text]* **inline** 절을 통해 입력된 데이터는 큰따옴표 또는 대괄호로 묶어야 합니다. 그 사이에 입력되는 텍스트는 파일의 내용과 동일한 방식으로 해석됩니다. 그러므로 텍스트 파일에 새 줄을 삽입하는 경우 **inline** 절의 텍스트에도 스크립트를 입력하면서 Enter 키를 누르는 등의 방법으로 동일하게 새 줄을 삽입해야 합니다. 첫 번째 줄에 열 수를 정의합니다. *format-spec ::= (fspec-item {, fspec-item })* 서식 사양은 괄호 안에 포함된 여러 서식 사양 항목 목록으로 구성됩니다. 자세한 내용은 *서식 사양 항목 (page 162)*를 참조하십시오.
resident	**resident**는 이전에 로드한 테이블에서 데이터를 로드해야 하는 경우에 사용됩니다. *table label*은 원래 테이블을 만든 **LOAD** 문 앞에 오는 레이블입니다. 이 레이블은 콜론으로 끝나야 합니다.

인수	설명
extension	분석 연결에서 데이터를 로드할 수 있습니다. 서버 측 확장(SSE) 플러그인에 정의된 함수를 호출하거나 스크립트를 평가하려면 **extension** 절을 사용해야 합니다. 단일 테이블을 SSE 플러그인에 보내면 단일 데이터 테이블이 반환됩니다. 플러그인이 반환되는 필드의 이름을 지정하지 않으면 필드의 이름이 Field1, Field2로 지정되는 식으로 계속됩니다. `Extension pluginname.functionname( tabledescription );` • SSE 플러그인의 함수를 사용하여 데이터 로드 *tabledescription ::= (table { ,tablefield})* 테이블 필드를 명시하지 않으면 필드가 로드 순서로 사용됩니다. • SSE 플러그인에서 스크립트를 평가하여 데이터 로드 *tabledescription ::= (script, table { ,tablefield})* **테이블 필드 정의에서 데이터 유형 처리** 데이터 유형은 분석 연결에서 자동으로 감지됩니다. 데이터에 숫자 값과 적어도 하나의 NULL이 아닌 텍스트 문자열이 있으면 이 필드가 텍스트로 간주됩니다. 다른 경우에는 숫자로 간주됩니다. **String()** 또는 **Mixed()**로 필드 이름을 래핑하여 데이터 유형을 강제로 설정할 수 있습니다. • **String()**은 필드를 텍스트로 강제 설정합니다. 필드가 숫자인 경우 이중 값의 텍스트 부분이 추출되고 변환은 수행되지 않습니다. • **Mixed()**는 필드를 이중으로 강제 설정합니다. **String()** 또는 **Mixed()**는 **extension** 테이블 필드 정의 외부에서 사용할 수 없으며 테이블 필드 정의에서 다른 Qlik Sense 함수를 사용할 수 없습니다.
where	**where**는 레코드를 선택에 포함할지를 나타내는 데 사용하는 절입니다. *criterion*이 True인 경우 선택 내용이 포함됩니다. *criterion*은 논리 표현식입니다.
while	**while**은 레코드를 반복적으로 읽어야 할지 여부를 지정하는 데 사용되는 절입니다. *criterion*이 True이면 동일한 레코드를 읽습니다. **while** 절을 유용하게 사용하려면 일반적으로 **IterNo( )** 함수를 포함해야 합니다. *criterion*은 논리 표현식입니다.
group by	**group by**는 데이터를 집계(그룹화)해야 하는 필드를 정의하는 데 사용되는 절입니다. 집계 필드는 로드한 표현식에 어떤 방식으로든 포함되어야 합니다. 집계 필드 외의 필드는 로드한 표현식의 집계 함수 외부에서 사용할 수 없습니다. *groupbyfieldlist ::= (fieldname { ,fieldname })*

인수	설명
order by	**order by**는 상주 테이블의 레코드가 **load** 문에 의해 처리되기 전에 정렬하는 데 사용되는 절입니다. 상주 테이블은 하나 이상의 필드를 기준으로 오름차순 또는 내림차순으로 정렬할 수 있습니다. 기본적으로 숫자 값을 기준으로 정렬되며, 2차적으로 국가별 정렬 순서에 따라 정렬됩니다. 이 절은 데이터 소스가 상주 테이블인 경우에만 사용할 수 있습니다. 정렬 필드는 상주 테이블의 정렬 기준 필드를 지정합니다. 이 필드는 상주 테이블 내의 이름 또는 번호(첫 번째 필드는 1번)로 지정할 수 있습니다. orderbyfieldlist ::= fieldname [sortorder] { , fieldname [sortorder] } *sortorder*는 *asc*(오름차순) 또는 *desc*(내림차순)입니다. *sortorder*를 지정하지 않으면 *asc*가 사용됩니다. *fieldname*, *path*, *filename* 및 *aliasname*은 각 이름의 의미를 나타내는 텍스트 문자열입니다. 소스 테이블의 모든 필드를 *fieldname*으로 사용할 수 있습니다. 그러나 as 절(*aliasname*)을 통해 만든 필드는 해당되지 않으며 같은 **load** 문에서 사용할 수 없습니다.

Let

let 문은 스크립트 변수 정의에 사용된다는 점에서 **set** 문과 차이가 있습니다. **let** 문은 **set** 문과는 반대로 변수에 할당하기 전 스크립트 런타임에 '='의 오른쪽에 있는 표현식을 평가합니다.

구문:

```
Let variablename=expression
```

예 및 결과:

예	결과

`Set x=3+4;`
 	
`Let y=3+4;`
 	
`z=$(y)+1;`
	`$(x)`는 '3+4'로 평가됩니다.
	
	`$(y)`는 ' 7'로 평가됩니다.
	
	`$(z)`는 ' 8'로 평가됩니다.
	
	Set 및 **Let** 문 사이의 차이점에 유의하십시오. **Set** 문은 문자열 '3+4'를 변수에 할당하는 반면 **Let** 문은 문자열을 평가하고 변수에 7을 할당합니다.

Let T=now( );	`$(T)`에는 현재 시간 값이 지정됩니다.

Set

set 문은 스크립트 변수를 정의하는 데 사용됩니다. 문자열, 경로, 드라이브 등을 대체하는 데 사용할 수 있습니다.

구문:

```
Set variablename=string
```

Example 1:

```
Set FileToUse=Data1.csv;
```

Example 2:

```
Set Constant="My string";
```

Example 3:

```
Set BudgetYear=2012;
```

Put

put 문은 하이퍼큐브에서 일부 숫자 값을 설정하는 데 사용됩니다.

열에 대한 액세스는 레이블로 수행할 수 있습니다. 선언 순서에 따라 열과 행에 액세스할 수도 있습니다. 자세한 내용은 아래 예를 참조하십시오.

구문:

```
put column(position)=value
```

Example 1:

열에 대한 액세스는 레이블로 수행할 수 있습니다.

이 예에서는 *Sales*라는 열의 첫 번째 위치에 값 `1`을 설정합니다.

```
Put Sales(1) = 1;
```

Example 2:

측정값에 `#hc1.measure` 형식을 사용하여 선언 순서별로 측정값 열에 액세스할 수 있습니다.

이 예는 최종 정렬된 하이퍼큐브의 10번째 위치에 값 1000을 설정합니다.

```
Put #hc1.measure.2(10) = 1000;
```

Example 3:

차원에 `#hc1.dimension` 형식을 사용하여 선언 순서에 따라 차원 행에 액세스할 수 있습니다.

이 예에서는 세 번째로 선언된 차원의 다섯 번째 행에 상수 Pi의 값을 넣습니다.

```
Put #hc1.dimension.3(5) = Pi();
```

값이나 레이블에 이러한 차원이나 표현식이 없으면 열을 찾을 수 없다는 오류가 반환됩니다. 열의 인덱스가 범위를 벗어나면 오류가 표시되지 않습니다.

HCValue

HCValue 함수는 지정된 열의 행에서 값을 검색하는 데 사용됩니다.

구문:

```
HCValue(column,position)
```

Example 1:

이 예에서는 레이블이 'Sales'인 열의 첫 번째 위치에 있는 값을 반환합니다.

```
HCValue(Sales,1)
```

Example 2:

이 예는 정렬된 하이퍼큐브의 10번째 위치에 있는 값을 반환합니다.

```
HCValue(#hc1.measure2,10)
```

Example 3:

이 예에서는 세 번째 차원의 다섯 번째 행에 있는 값을 반환합니다.

```
HCValue(#hc1.dimension.3,5)
```

값이나 레이블에 이러한 차원이나 표현식이 없으면 열을 찾을 수 없다는 오류가 반환됩니다. 열의 인덱스가 범위를 벗어나면 NULL이 반환됩니다.

11 Qlik Sense에서 지원되지 않는 QlikView 함수 및 문

QlikView 로드 스크립트와 차트 표현식에서 사용할 수 있는 대부분의 함수 및 문은 Qlik Sense에서도 지원되지만 여기에 설명하는 일부 예외가 있습니다.

11.1 Qlik Sense에서 지원되지 않는 스크립트 문

Qlik Sense에서 지원되지 않는 QlikView 스크립트 문

문	주석
Command	대신 **SQL**을 사용하십시오.
InputField	

11.2 Qlik Sense에서 지원되지 않는 함수

이 목록은 Qlik Sense에서 지원되지 않는 QlikView 스크립트 및 차트 함수를 설명합니다.

- **GetCurrentField**
- **GetExtendedProperty**
- **Input**
- **InputAvg**
- **InputSum**
- **MsgBox**
- **NoOfReports**
- **ReportComment**
- **ReportId**
- **ReportName**
- **ReportNumber**

11.3 Qlik Sense에서 지원되지 않는 접두사

이 목록은 Qlik Sense에서 지원되지 않는 QlikView 접두사를 설명합니다.

- **Bundle**
- **Image_Size**
- **Info**

12 에서 권장되지 않는 함수 및 문Qlik Sense

QlikView 로드 스크립트 및 차트 표현식에서 사용할 수 있는 대부분의 함수 및 문은 Qlik Sense에서도 지원되지만 그 중 일부는 Qlik Sense에서 사용하는 것이 권장되지 않습니다. 사용이 중단된 이전 버전의 Qlik Sense에서 사용 가능한 함수 및 문도 있습니다.

이들 함수 및 문은 여전히 의도한 대로 작동하지만 호환성 문제로 다음 버전에서는 제거될 수도 있으므로 이 섹션의 권장 사항에 따라 코드를 업데이트하는 것이 좋습니다.

12.1 Qlik Sense에서 권장되지 않는 스크립트 문

다음 표에는 Qlik Sense에서 사용이 권장되지 않는 스크립트 문이 포함되어 있습니다.

권장되지 않는 스크립트 문

문	권장 사항
Command	대신 **SQL**을 사용하십시오.
CustomConnect	대신 **Custom Connect**을 사용하십시오.

12.2 Qlik Sense에서 권장되지 않는 스크립트 문 매개 변수

다음 표에서는 Qlik Sense에서 사용이 권장되지 않는 스크립트 문 매개 변수를 설명합니다.

권장되지 않는 스크립트 문 매개 변수

문	매개 변수
Buffer	대신 **Incremental**을 사용하십시오. • **Inc**(권장되지 않음) • **Incr**(권장되지 않음)

문	매개 변수
LOAD	다음 매개 변수 키워드는 QlikView 파일 변환 마법사에서 생성됩니다. 데이터를 다시 로드하면 기능은 유지되지만 Qlik Sense에서는 이 매개 변수를 사용하는 문 생성을 위한 가이드 지원/마법사를 제공하지 않습니다. • **Bottom** • **Cellvalue** • **Col** • **Colmatch** • **Colsplit** • **Colxtr** • **Compound** • **Contain** • **Equal** • **Every** • **Expand** • **Filters** • **Intarray** • **Interpret** • **Length** • **Longer** • **Numerical** • **Pos** • **Remove** • **Rotate** • **Row** • **Rowend** • **Shorter**

12.3 Qlik Sense에서 권장되지 않는 함수

다음 표에서는 Qlik Sense에서 사용이 권장되지 않는 스크립트 및 차트 함수를 설명합니다.

권장되지 않는 함수

함수	권장 사항
NumAvg **NumCount** **NumMax** **NumMin** **NumSum**	대신 범위 함수를 사용하십시오. *범위 함수 (page 1282)*
Color() **QliktechBlue** **QliktechGray**	대신 다른 색 함수를 사용하십시오. **QliktechBlue()**를 **RGB(8, 18, 90)**으로 바꾸고, **QliktechGray**를 **RGB(158, 148, 137)**로 바꾸면 동일한 색을 얻을 수 있습니다. *색 함수 (page 530)*
QlikViewVersion	대신 **EngineVersion**을 사용하십시오. *EngineVersion (page 1424)*
ProductVersion	대신 **EngineVersion**을 사용하십시오. *EngineVersion (page 1424)*
QVUser	
Year2Date	대신 **YearToDate**을 사용하십시오.
Vrank	대신 **Rank**을 사용하십시오.
WildMatch5	대신 **WildMatch**을 사용하십시오.

ALL 한정자

QlikView에서는 **ALL** 한정자가 표현식 앞에 올 수 있습니다. 이는 **{1} TOTAL**를 사용하는 것과 동일합니다. 그런 경우, 차트 차원 및 현재 선택에 관계없이 문서 내 필드의 모든 값에 대해 계산이 이루어집니다. 문서의 논리 상태에 관계없이 항상 동일한 값이 반환됩니다. **ALL** 한정자를 사용하는 경우, **ALL** 한정자가 자체적으로 집합을 정의하므로 집합 표현식을 사용할 수 없습니다. 앞서 말한 이유로, **ALL** 한정자는 이 버전의 Qlik Sense에서도 여전히 작동하지만 향후 버전에서 제거될 수 있습니다.